UNIVERSITY OF TEXAS ARLINGTON

Project Report
Web and social analytics INSY-5337-001

SENTIMENT ANALYSIS and END TO END PIPELINEOF CHATGPT RELATED TWEETS

GROUP - 3
JAY HITESHKUMAR PATEL (ID-1002118032)
KAJAL SATISHKUMAR DUBEY (ID-1002110756)
KHUSHBOO HITESH PAREKH (ID-1002117550)
MONIKA CHANDRAKANT INGALE (ID-1002120701)
ROHAN GOIEIT (ID-1002083318)
SPURTHI CHEEDARLA (ID-1002111941)

UNDER THE GUIDANCE OF:
DR. RIYAZ SIKORA

Table of Contents

1. ABSTRACT

ChatGPT is a chatbot that uses a powerful language model (GPT4) to have conversations with users. It has become very popular and has a lot of users. It is known for its ability to generate language and perform various tasks using text. However, some people are concerned about how it might affect society.

In this project, we try to build end to end pipeline from extracting dataset from twitter to deploying the model in aws sagemaker. We used python nltk library for sentiment analysis and neural network for training and future prediction of sentiment of tweet about ChatGPT. Our most of time was consumed in technical study of building the ETL pipeline. Further, With growing need of AI, we need huge computational power to train large and complex model. As building inhouse system to train these models require loads of effort. Hence, cloud platform can save us huge cost and headache. So, we tried to build framework, tools and steps needed to deploy model on AWS platform. This was whole new learning experience for us as we looked project from end to end machine learning cycle, instead of just data visualization and prediction.

The results of our sentiment analysis showed that overall, people's feelings towards ChatGPT were mostly neutral to positive. This sentiment was consistent across different occupation groups, meaning that people from different professions generally had similar opinions.

We also found that people talked about a wide range of topics in their tweets. The most popular topics mentioned were ChatGPT, Artificial Intelligence etc. These were the subjects that people seemed to be most interested in when discussing ChatGPT on Twitter. By conducting this research, we gained insights into what people think about ChatGPT. This helps us better understand its impact on society and the general attitudes towards this type of language model.

2. BACKGROUND OF THE STUDY

ChatGPT, an AI chatbot created by OpenAI and introduced in November 2022, utilizes OpenAI's GPT-4 series of extensive language models. Through a combination of supervised and reinforcement learning methods, the chatbot has been refined, a technique known as transfer learning. The remarkable advantages offered by ChatGPT compared to conventional chatbots have resulted in its rapid adoption, surpassing the user count of popular online platforms like Netflix, Facebook, and Instagram. Within just five days of its launch, ChatGPT attracted over 1 million users, and within two months, it reached an impressive user base of 100 million. Early adopters of ChatGPT speculate that it may eventually render certain content creation professions obsolete. ChatGPT has demonstrated its capability to generate high-quality responses across various challenges, including solving coding problems and providing accurate answers to examination queries.

Figure 1: Time taken Reach 100M Users

3. INTRODUCTION

ChatGPT has gained immense popularity online and has quickly amassed 100 million monthly active users in just 2 months, surpassing other major platforms. Developed by OpenAI, ChatGPT is a powerful language model that functions as a chatbot, generating realistic text for various purposes such as academic writing, language translations, and coding. It has the potential to transform various fields and increase human productivity by automating repetitive tasks. However, ChatGPT has limitations and challenges that need to be addressed.

Despite the widespread discussions surrounding ChatGPT, there is limited research on public attitudes towards it. Our study aims to fill this gap by analyzing sentiments expressed in tweets

related to ChatGPT and identifying the main topics discussed. We will also explore how people from different professions perceive and discuss ChatGPT. By conducting sentiment analysis, topic modeling, and analyzing tweets based on occupations, we aim to gain a comprehensive understanding of people's opinions and attitudes towards ChatGPT.

Basically, our study will delve into the sentiment, topics, and occupational perspectives found in ChatGPT-related tweets. This analysis will provide a more nuanced understanding of public attitudes towards ChatGPT and shed light on the diverse discussions and perspectives surrounding this powerful language model.

4. OBJECTIVE

By employing a combination of methodologies, examine tweets posted between April 2023 and May 2023 that discuss ChatGPT and encompass a wide range of unstructured viewpoints. Explore the prevailing themes and sentiments within these discussions while also investigating the perspectives of early ChatGPT users. Through this process, our main goal is to see how end to end machine learning project looks like and difficulty in production. As to train complex model, we need computational capability and building one in house would be costly matter. Therefore, cloud platform can make our life easier in building end to end pipeline.

5. OVERVIEW OF SENTIMENT ANALYSIS:

Sentiment analysis is a natural language processing (NLP) technique that aims to understand the sentiment or emotional tone expressed in a piece of text. Its primary goal is to categorize the text as positive, negative, or neutral based on the emotions conveyed by the words used. This technology has become increasingly important as text data continues to grow exponentially, especially on social media platforms and in customer interactions.

6. THE PROCESS OF SENTIMENT ANALYSIS INVOLVES SEVERAL KEY STEPS:

- **Preprocessing:** This step involves cleaning and preparing the text data for analysis. It may include tasks like removing punctuation, converting all text to lowercase, handling contractions, and eliminating stop words (commonly used words with little semantic meaning).
- **Feature Extraction:** In this step, relevant features or characteristics are extracted from the preprocessed text. These features could be individual words, word combinations (n-grams), or other linguistic elements that can be used to represent the sentiment of the text.
- **Sentiment Classification:** The extracted features are then used to classify the sentiment of the text. There are different approaches for this classification, ranging from simple rule-based methods to more sophisticated machine learning algorithms.
- **Evaluation:** After sentiment classification, the accuracy and effectiveness of the sentiment analysis model are evaluated. This step helps ensure the model's performance and identify areas for improvement.

7. VARIOUS TECHNIQUES ARE EMPLOYED IN SENTIMENT ANALYSIS:

a. Rule-based methods: These methods use predefined rules and patterns to determine sentiment. For example, specific words or phrases may be assigned positive or negative scores, and the overall sentiment is computed based on these scores.

b. Lexicon-based methods: Lexicons are dictionaries that contain words with associated sentiment scores. These methods involve looking up the sentiment scores of words in the text and aggregating them to determine the overall sentiment.

c. Machine learning methods: Machine learning models are trained on labeled data, where each text sample is associated with its sentiment category. These models learn to generalize patterns from the training data and make predictions on new, unseen data.

8. THE APPLICATIONS OF SENTIMENT ANALYSIS ARE DIVERSE:

- **Social Media Analysis:** Companies use sentiment analysis to monitor their brand reputation and understand how customers perceive their products or services on social media platforms.
- **Market Research:** Sentiment analysis helps businesses gauge customer opinions about products and competitors, providing valuable insights for marketing strategies.
- **Customer Support:** Sentiment analysis can be used to automatically categorize customer feedback and identify areas of concern or dissatisfaction, enabling timely responses to customer issues.
- **Brand Monitoring:** Organizations track mentions of their brand in online discussions and reviews to assess public sentiment and address any negative publicity promptly.
- **Political Analysis:** Sentiment analysis is used in political contexts to understand public opinions on political candidates, policies, and events.

Overall, sentiment analysis plays a crucial role in decision-making by providing actionable insights into public sentiment. It enables businesses and organizations to better understand their audience, improve products and services, and enhance customer experiences.

9. THE FOLLOWING RESEARCH QUESTIONS WILL BE ANSWERED DURING THIS ANALYSIS:

1. How can we extract and transform Twitter data using an ETL pipeline to gain valuable insights into customer opinions and engagement?

2. What are the distribution patterns of likes, retweets, and replies in our Twitter data, and how can we leverage this information to understand customer engagement levels?

3. How can we preprocess and extract meaningful features from Twitter data to uncover key trends, topics, or sentiments discussed by our customers?

4. What are the important words and phrases that stand out in the Twitter data, and how can we utilize NLP methods like TF-IDF and bag of words to highlight and analyze them?

5. How can we leverage machine learning algorithms neural network to predict the sentiment of text data in tweets and gain insights into customer sentiment on a larger scale?

6. How to make use of cloud platforms (AWS cloud in this case) for model deployment and making product live to customer using REST API

10. METHODOLOGY:

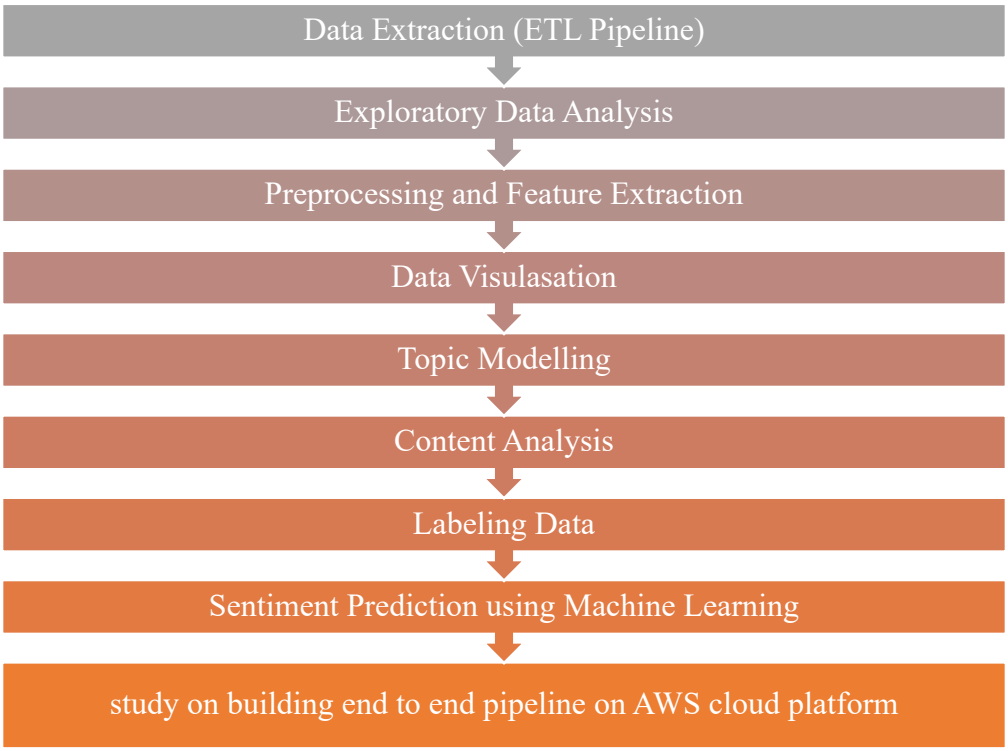


Figure 3: Methodology Flow chart

Table: Data processing and techniques

Task	Technique Description	Tools/Packages used
Data Collection	collected data from Kaggle	NA
Data Preprocessing	Duplication removal, lowercasing, noise removal (punctuation, URLs, @users)	Regularr expression, NLTK, pandas, numpy
Sentiment Analysis	Quantitative sentiment analysis of each topic via rule-based and deep learning-based model	NLTK VADER
Data Visualization	Multi-attribute plots	tableau, matplotlib, seaborn, wordcloud, PowerBI
Environments & Platforms	Python	Google Colab, Jupyter Notebook, Twitter, ChatGPT
Cloud Platform for Model deployment	AWS cloud	S3, AWS Glue, Lamda function, Amazon Groundtruth, sagemaker, REST API

Data Extraction from Twitter API and ETL pipeline is performed to retrieve and organize the relevant data. Exploratory data analysis involves plotting histograms to visualize the distribution of likes, retweets, and replies to tweets. Preprocessing and feature extraction techniques are applied to prepare the data for analysis. Content analysis includes highlighting important words using NLP methods such as TF-IDF and bag of words. Tonality analysis is conducted using TextBlob in Python, which involves tasks like spelling correction, part of speech tagging, and text classification. Tokenization is performed using the Natural Language Toolkit (NLTK) library.The data is labeled based on the tonality analysis results using the Label Encoder in Python.Sentiment prediction in tweets using ML: Neural network.

11. ETL PIPELINE (EXTRACT, TRANSFORM, LOAD) USING TWITTER API:

ETL stands for Extract, Transform, Load. It is a data integration process commonly used in the field of data warehousing and business intelligence to extract data from various sources, transform it into a consistent and usable format, and load it into a target data repository or database for analysis and reporting purposes.
Let's break down the three main stages of the ETL process:

Extract:

In the first step, data is extracted from multiple heterogeneous sources, which could include databases, spreadsheets, flat files, APIs, web services, and other data repositories. The sources might be located on premises or in the clouds. The extraction process involves querying the sources

and pulling the necessary data into the ETL system. This data is often raw and unstructured or may have different formats and data types.

Transform:

Once the data is extracted, it needs to be transformed to ensure it is consistent, clean, and usable. Data transformation involves a series of operations, such as data cleansing (removing duplicates, handling missing values), data integration (combining data from multiple sources), data enrichment (adding additional information), data aggregation (summarizing data), and data format conversion (e.g., converting date formats). The transformation process ensures that the data is standardized and suitable for the target data repository or data warehouse.

Load:

After the data has been extracted and transformed, it is loaded into the target data warehouse or database. The load phase involves populating the data into the appropriate tables or data structures in the destination system. Depending on the architecture and requirements, the data can be inserted, updated, or merged into the target database. The loaded data is now ready for querying and analysis, enabling business users to generate reports, perform data analytics, and gain valuable insights for decision-making.

The **ETL process** is essential in data integration and management, especially when dealing with large volumes of data from diverse sources. It ensures that data is prepared and organized in a consistent manner, allowing organizations to have a unified and reliable view of their data for analysis and reporting purposes. Additionally, ETL plays a crucial role in maintaining data quality, as it helps identify and correct errors, inconsistencies, and discrepancies in the data during the transformation phase.

First, we Build a connection with Twitter's data by using a special key called a "bearer token." This key allows the script to access and interact with Twitter's information.

Secondly, we first must decide what we are looking for. As we are doing analysis of chatgpt related tweets, the script will consider only those criteria's which we have specifically mentioned and avoids showing the remaining.

Next, we are using a function called "search tweets", that uses Twitter's API to search for tweets that match the criteria we set earlier.

In **Printing the first tweet** step, the criteria's we are searching by using the search tweet function will print the text of the first tweet it finds.

After this the script creates a kind of table, called a DataFrame, where it will store the information, it finds. "Search tweets" function searches for tweets from each hour of a specific day. Then it filters those tweets based on certain rules, like checking if they have sensitive content or too many hashtags, mentions, or web links. After filtering, it keeps the details of the remaining tweets in the DataFrame for further analysis.

Lastly, for each tweet that passes the filters, the script saves important information like the tweet's ID, when it was created, the actual text of the tweet, what language it's in, and what device or app the user used to post it. It also saves details about the person who posted the tweet, like their user ID, name, username, location, and a brief description about them. Additionally, it keeps track of the number of followers, following, and tweets that person has.

12. DATA DESCRIPTION:

The dataset being used is the 'ChatGPT 1000 Daily Tweets Dataset.' It contains information about tweets related to ChatGPT, including tweet ID, creation and extraction timestamps, text content, user details, language, engagement metrics, and more. This dataset enables analysis and exploration of user interactions and sentiments regarding the ChatGPT model on Twitter. We had extracted 450 tweets from Twitter by using API and remaining from the kaggle.

https://www.kaggle.com/code/edomingo/etl-chatgpt-1000-daily-tweets-dataset

So, in the data description we have mentioned the number of rows, columns & duration of original dataset.

Original Dataset

Number of Rows	Number of Columns	Date
42003	20	April 3, 2023, to May 11, 2023

Figure 4: Kaggle dataset format

Variables present in the Original Dataset:

tweet_id: A unique identifier assigned to each individual tweet, used to distinguish one tweet from another.
tweet created: The timestamp indicating when the tweet was originally created or posted.
tweet extracted: The timestamp indicating when the tweet was extracted or collected from the source platform.
text: The actual content of the tweet, which includes the message or text posted by the user.
lang: The language in which the tweet is written or posted. It represents the language code or abbreviation (e.g., "en" for English, "es" for Spanish).
user_id: A unique identifier associated with the user who posted the tweet.
user_name: The display name or username of the user who posted the tweet.
user_username: The unique handle or username of the user who posted the tweet, often preceded by the "@" symbol.
user_location: The location or geographic information provided by the user in their profile.
user_description: The user's profile description or bio, which may contain additional information about the user.
user_created: The timestamp indicating when the user's account was created.
user_followers_count: The number of followers the user has at the time of the tweet.
user_following_count: The number of accounts the user is following at the time of the tweet.
user_tweet_count: The total count of tweets posted by the user up to the time of the current tweet.
user_verified: A binary indicator (e.g., True/False) showing whether the user's account is verified by the platform (e.g., Twitter).

source: The application or platform from which the tweet was posted, often represented as the source URL or application name.
retweet_count: The number of times the tweet has been retweeted by other users.
like_count: The number of times the tweet has been liked or favorited by other users.
reply_count: The number of replies or responses the tweet has received from other users.
impression_count: The total number of times the tweet has been viewed or displayed to users, indicating its reach or visibility.

13. DATA-PREPROCESSING

Imported necessary libraries:

```
#Installing necessary libraries

!pip install nltk

# Import Data Preprocessing and Wrangling libraries
import re
from tqdm.notebook import tqdm
import pandas as pd
import numpy as np
from datetime import datetime
import dateutil.parser

#Initialization
import os

from nltk.corpus import stopwords
nltk.download('stopwords')

!pip install pyspellchecker
!pip install scattertext
!pip install nltk
!pip install -U kaleido
!pip install wordcloud

# Import NLP Libraries
import nltk
from spellchecker import SpellChecker
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA

# Import Visualization Libraries
import plotly.offline as pyo
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import seaborn as sns
import scattertext as st
import scattertext as st
from IPython.display import IFrame
from wordcloud import WordCloud, ImageColorGenerator
import random

# Downloading periphrals
nltk.download('vader_lexicon')

import warnings
warnings.filterwarnings('ignore')
```

Figure 5:

Loading Kaggle ChatGPT-tweets' Dataset:

```
#Initialization

import os
directory = 'C:\\Users\\waghm\\Downloads\\chatgpt_daily_tweets.csv'
csv_files = [file for file in os.listdir(directory) if file.endswith('.csv')]
df_list = []
for csv in csv_files:
    df = pd.read_csv(os.path.join(directory, csv))
    df_list.append(df)
```

```
df.info()
```

Figure 6: Kaggle chatGPT-Tweets Dataset

14. DATA CLEANING

```
data = df.copy()
data['original_tweet'] = df['text']
data['datetime'] = data['tweet_created']
data['datetime'] = data.datetime.apply(lambda x: dateutil.parser.parse(x))
rt_mask = data.text.apply(lambda x: "RT @" in x)

# standard tweet preprocessing
data.text = data.text.str.lower()
#Remove twitter handlers
data.text = data.text.apply(lambda x:re.sub('@[^\s]+','',x))
#remove hashtags
data.text = data.text.apply(lambda x:re.sub(r'\B#\S+','',x))
# Remove URLS
data.text = data.text.apply(lambda x:re.sub(r"http\S+", "", x))
# Remove all the special characters
data.text = data.text.apply(lambda x:' '.join(re.findall(r'\w+', x)))
#remove all single characters
data.text = data.text.apply(lambda x:re.sub(r'\s+[a-zA-Z]\s+', '', x))
# Substituting multiple spaces with single space
data.text = data.text.apply(lambda x:re.sub(r'\s+', ' ', x, flags=re.I))

# convert the 'date' column to datetime format and remove the timezone information
data['datetime'] = pd.to_datetime(data['datetime']).dt.tz_localize(None)


# Viewing the preprocessed data
data.head()
```

Figure 7: Creating a dataframe

Creating a copy of the dataframe: We make a copy of the original dataframe 'df' and assign it to a new variable called 'data'.

Copying columns: We create three new columns in the 'data' dataframe. The 'original_tweet' column is created by copying the values from the 'text' column of the 'df' dataframe. The 'datetime' column is created by copying the values from the 'tweet_created' column of the 'data' dataframe.

Parsing datetime values: We convert the values in the 'datetime' column to datetime format using the 'dateutil.parser.parse' function.

Creating a mask for retweets: We create a Boolean mask called 'rt_mask' by checking if each tweet in the 'text' column contains the string "RT @".

Standard tweet preprocessing: We convert all the text in the 'text' column to lowercase using the 'str.lower()' function.

Removing Twitter handles: We remove Twitter handles (user mentions starting with '@') from the text in the 'text' column using regular expressions and the 're.sub()' function.

Removing hashtags: We remove hashtags (words starting with '#') from the text in the 'text' column using regular expressions and the 're.sub()' function.

Removing URLs: We remove URLs (strings starting with "http" and followed by any non-whitespace characters) from the text in the 'text' column using regular expressions and the 're.sub()' function.

Removing special characters: We remove all special characters from the text in the 'text' column using regular expressions and the 're.findall()' function to extract alphanumeric characters.

Removing single characters: We remove all single characters (standalone alphabetic characters) from the text in the 'text' column using regular expressions and the 're.sub()' function.

Substituting multiple spaces: We substitute multiple spaces in the text with a single space using regular expressions and the 're.sub()' function.

Converting 'datetime' column: We convert the values in the 'datetime' column to datetime format using the 'pd.to_datetime()' function and then remove the timezone information using the 'dt.tz_localize(None)' method.

Viewing the preprocessed data: We display the first few rows of the preprocessed 'data' dataframe using the 'head()' function.

```
df_en = df[df['lang']=='en'].reset_index(drop=True)
df_en.shape
```

```
(20114, 19)
```

```
df_en = df_en.drop(columns = ['lang'])
```

```
#Remove URL from tweet text
df_en['text'] = df_en['text'].apply(lambda x: re.sub(r'http\S+', '',x))
#Remove mention (@user)
df_en['text'] = df_en['text'].apply(lambda x: re.sub(r'@\w+', '',x))
#All lowercases
df_en['text'] = df_en['text'].apply(lambda x: ' '.join(x.lower() for x in x.split()))
#Remove Punctuation
df_en['text'] = df_en['text'].apply(lambda x: re.sub('[^\w\s]', '', x))
```

```
#Replace 'chat GPT' with 'chatGPT'
df_en['text'] = df_en['text'].apply(lambda x: re.sub(r'chat GPT', 'chatGPT', x))
```

```
df_en
```

Figure 8: preprocessed data

Removing URLs from tweet text: We use the 're.sub()' function with a regular expression (r'http\S+') to remove any URLs (strings starting with "http" and followed by any non-whitespace characters) from the 'text' column of the 'df_en' dataframe. The resulting text is then assigned back to the 'text' column.
Removing mentions: We use the 're.sub()' function with a regular expression (r'@\w+') to remove mentions (strings starting with '@' followed by alphanumeric characters) from the 'text' column of the 'df_en' dataframe. The resulting text is then assigned back to the 'text' column.
Lowercasing: We convert all the text in the 'text' column to lowercase by splitting each tweet into individual words using the 'split()' method, converting each word to lowercase using the 'lower()' method, and then joining the words back together with a space in between. The updated text is assigned back to the 'text' column.
Removing punctuation: We use the 're.sub()' function with a regular expression ('[^\w\s]') to remove any punctuation (non-alphanumeric characters except spaces) from the 'text' column of the 'df_en' dataframe. The resulting text is then assigned back to the 'text' column.
Replacing 'chat GPT' with 'chatGPT': We use the 're.sub()' function with the regular expression (r'chat GPT') to replace the occurrence of 'chat GPT' with 'chatGPT' in the 'text' column of the 'df_en' dataframe. The resulting text is then assigned back to the 'text' column.
These preprocessing steps help in cleaning and standardizing the tweet text in the 'text' column of the 'df_en' dataframe, making it more suitable for further analysis or natural language processing tasks.

To ensure the accuracy of our analysis, we implemented various measures to eliminate unnecessary tweets from our dataset. These measures included:

- We removed duplicate tweets from the dataset.
- We filtered out tweets that were written in English and stored them in a new dataframe.
- We converted all tweets that represented the same word but in different cases (e.g., ChatGPT and CHATGPT) to a consistent lowercase form (e.g., chatgpt).
- We eliminated noise such as punctuation, URLs, and Twitter handles using the "re" library.
- Unwanted columns were removed.
- We performed standard preprocessing on the tweets.
- Twitter handles were removed.
- Hashtags were removed.
- URLs were removed.
- All special characters were removed.
- Single characters were removed.
- Multiple spaces were substituted with a single space.
- The 'date' column was converted to datetime format and the timezone information was removed.
- The CSV dataset was stored in a dataframe.

After implementing these preprocessing techniques, our cleaned dataset consisted of 20,114 unique entries, which were then ready for further analysis.

Dataset after cleaning:

Number of Rows	Number of Columns
20114	22

Figure 9: Cleaned dataset

Variables present in the cleaned Dataset:

tweet_id: A unique identifier assigned to each individual tweet, used to distinguish one tweet from another.
tweet_created: The timestamp indicating when the tweet was originally created or posted.
tweet_extracted: The timestamp indicating when the tweet was extracted or collected from the source platform.
text: The actual content of the tweet, which includes the message or text posted by the user.
lang: The language in which the tweet is written or posted. It represents the language code or abbreviation (e.g., "en" for English, "es" for Spanish).
user_id: A unique identifier associated with the user who posted the tweet.
user_name: The display name or username of the user who posted the tweet.
user_username: The unique handle or username of the user who posted the tweet, often preceded by the "@" symbol.
user_location: The location or geographic information provided by the user in their profile.
user_description: The user's profile description or bio, which may contain additional information about the user.
user_created: The timestamp indicating when the user's account was created.
user_followers_count: The number of followers the user has at the time of the tweet.
user_following_count: The number of accounts the user is following at the time of the tweet.

user_tweet_count: The total count of tweets posted by the user up to the time of the current tweet.
user_verified: A binary indicator (e.g., True/False) showing whether the user's account is verified by the platform (e.g., Twitter).
source: The application or platform from which the tweet was posted, often represented as the source URL or application name.
retweet_count: The number of times the tweet has been retweeted by other users.
like_count: The number of times the tweet has been liked or favorited by other users.
reply_count: The number of replies or responses the tweet has received from other users.
impression_count: The total number of times the tweet has been viewed or displayed to users, indicating its reach or visibility.
original_tweets: This column may contain a binary indicator or a count of how many original tweets are part of the data. An original tweet is a tweet created by the user and not a retweet of someone else's tweet.

date: The date associated with the tweet. This could be the date when the tweet was originally posted (tweet_created) or the date when it was extracted (tweet_extracted).

15. FEATURE ENGINEERING:

- The datetime column provides the timestamp of each tweet's creation. By extracting features like hour, date, month, and year from this column, we can study tweet patterns over different time periods. This helps us identify peak tweeting hours, daily or monthly trends, and long-term changes in tweet activity and sentiment.
- In short, fixing spelling mistakes in tweet content ensures that the analysis is based on correct and accurate text. **This step is crucial because misspelled words can lead to misinterpretations and inaccurate results.**
- On the other hand, sentiment analysis involves assessing the sentiment expressed in each tweet, determining whether it is positive, negative, or neutral. By understanding the overall sentiment of tweets, we can gain insights into how people feel about a particular topic, product, event, or any other subject of interest. This analysis helps businesses, researchers, and individuals to gauge public opinion and make informed decisions based on the sentiment expressed by Twitter users.

16. DATA VISUALIZATION:

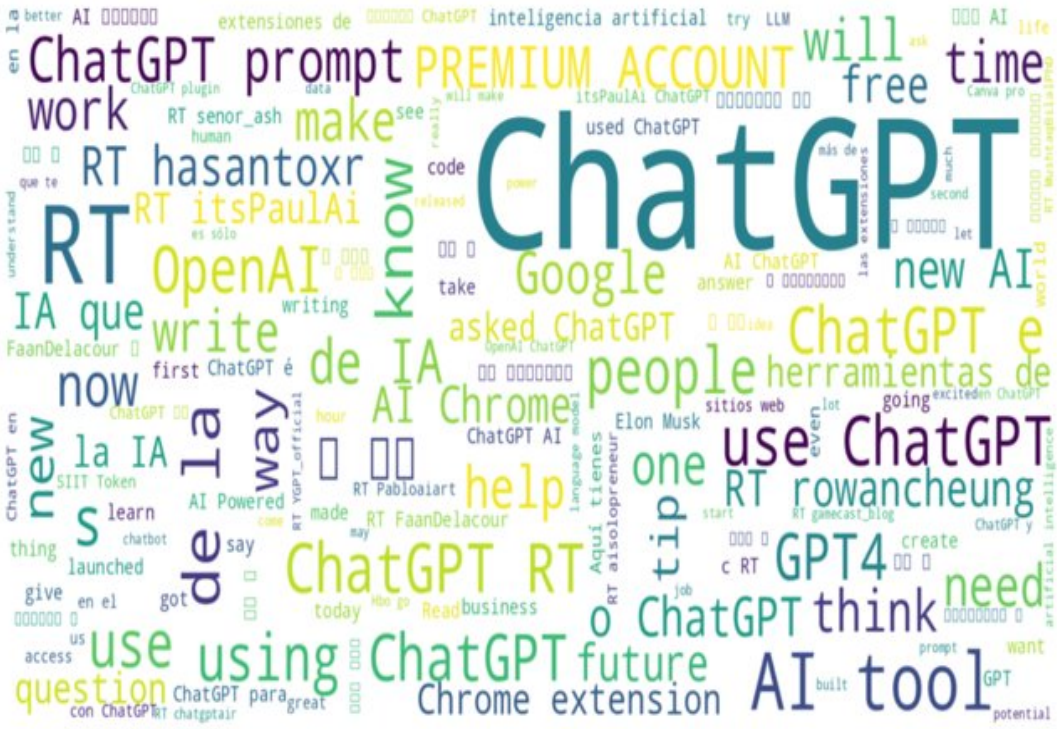


Figure 10: prevalent words in tweets

- Word clouds represent the frequency of words in the provided text data.
- More frequent words are displayed with larger and more prominent fonts in the word cloud.
- Word clouds help identify the most common or significant words in the text data.
- The size and prominence of words in the word cloud convey their importance and usage frequency.
- Word clouds provide a visual and intuitive way to understand the content and identify key themes.
- They are helpful in summarizing large amounts of text and gaining quick insights.
- Word clouds are widely used in various fields, including data analysis, content marketing, and social media analysis.
- They are engaging and visually appealing, making them suitable for presentations and data visualization.

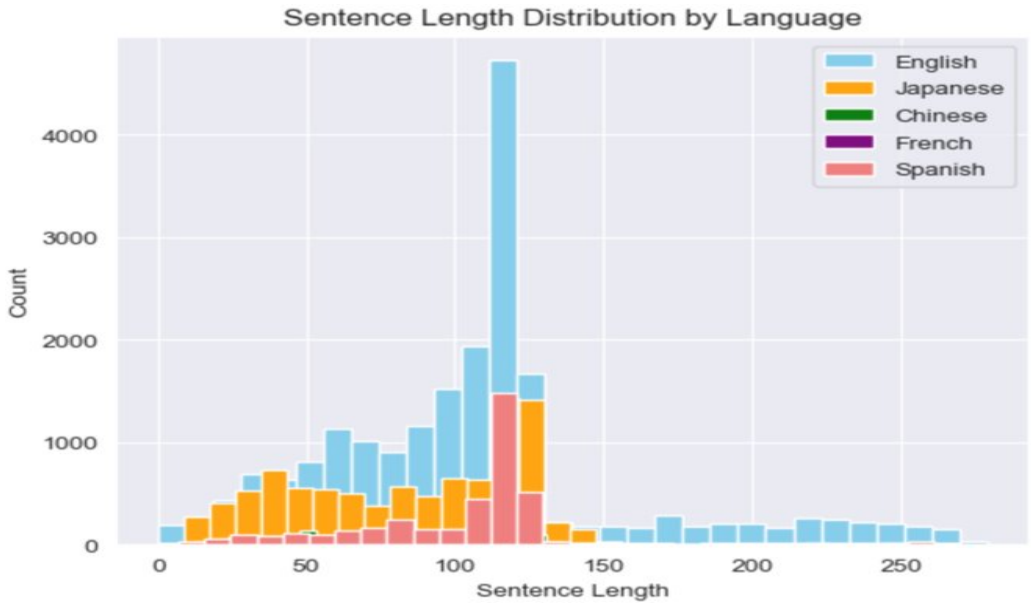


Figure 11: Sentence Length Distribution by Language

- We create separate histograms for English, Japanese, Chinese, French, and Spanish sentence lengths.
- This allows us to identify patterns or differences in sentence lengths among these languages.
- The histograms are valuable tools for us in natural language processing, translation, and cross-linguistic studies.
- They support us in making data-driven decisions in language-related tasks.
- By analyzing the histograms, we gain insights into the distribution of sentence lengths in each language, aiding in language understanding and analysis.

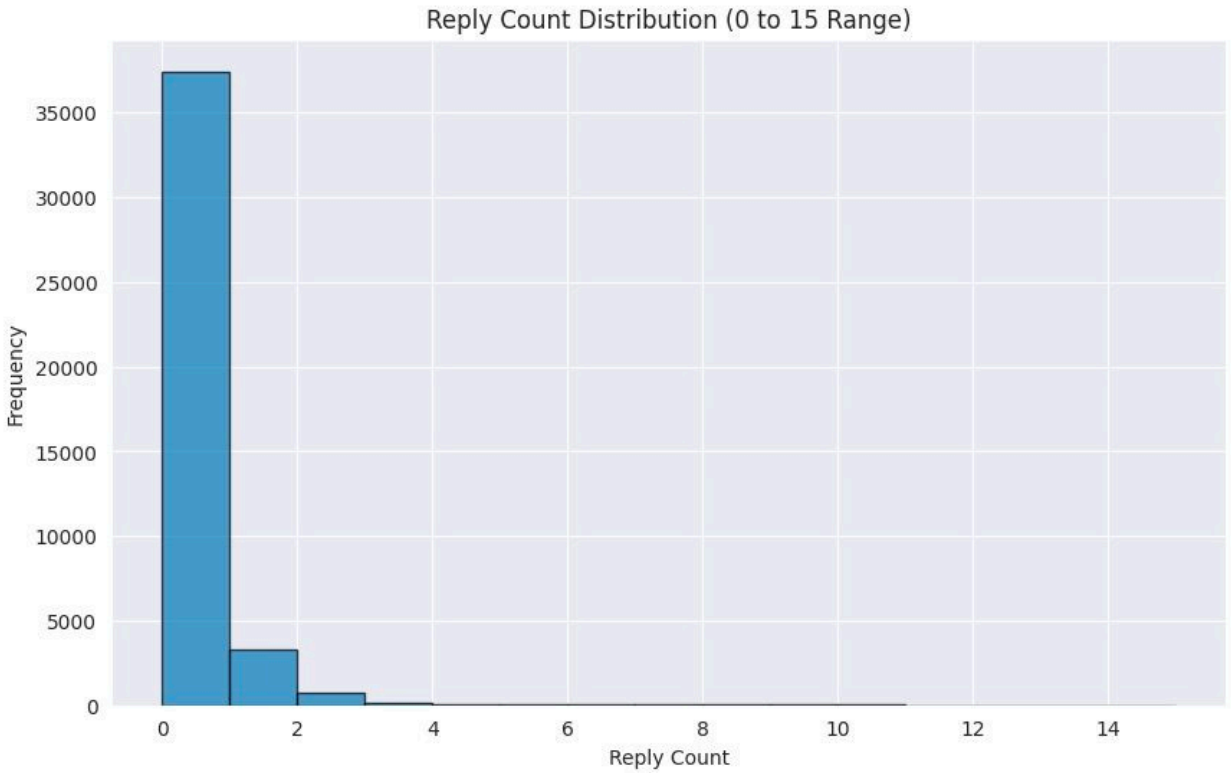


Figure 12: Reply count distribution on no of tweets

- The plot is a histogram representing the distribution of the number of responses (reply count) in a dataset of tweets.
- The histogram divides the reply count range into 50 bins for visualization.
- The x-axis is labeled as "Reply count," showing the number of responses received by tweets.
- The y-axis is labeled with the corresponding number of tweets falling within each reply count interval.
- The histogram facilitates easy analysis of tweet concentration across different reply count intervals.
- It helps in identifying trends, patterns, and potential outliers in the data. The histogram is valuable for understanding engagement levels and identifying popular tweets.
- It provides insights into the dynamics of social media interactions and how tweets receive varying numbers of responses.

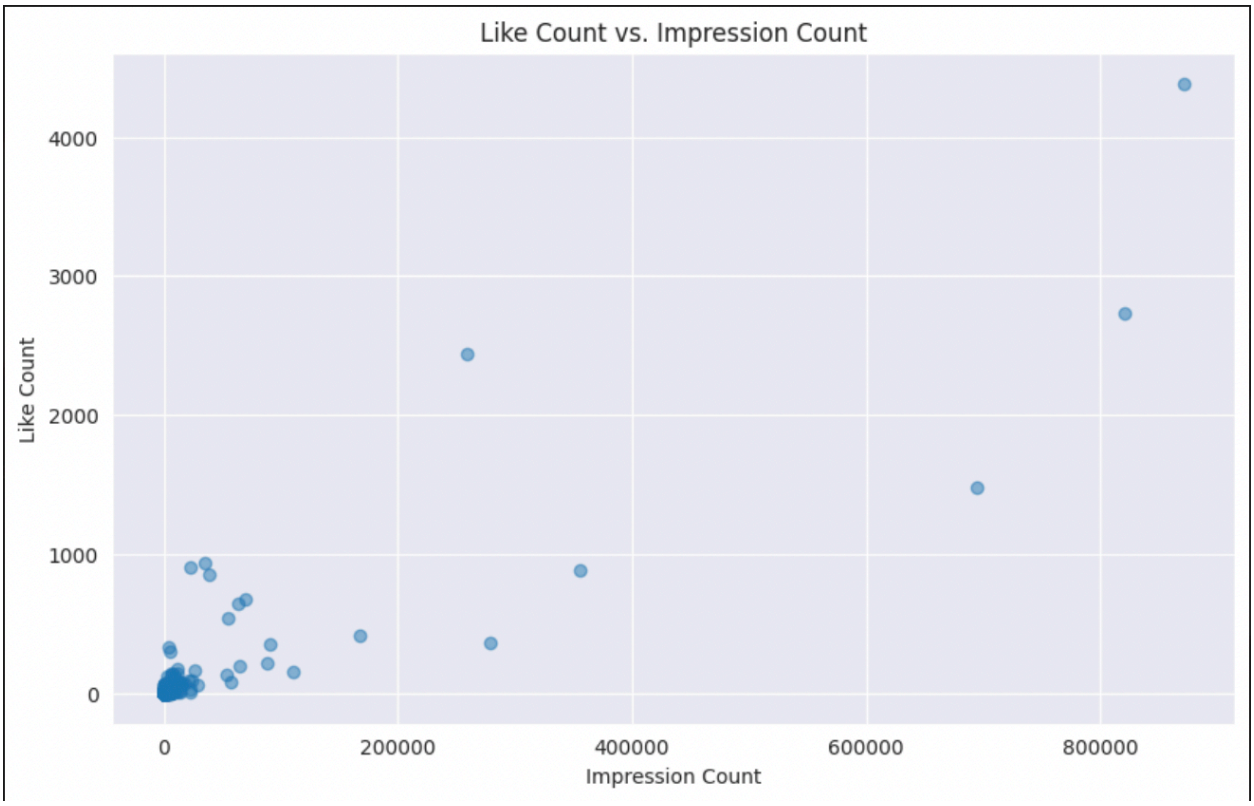


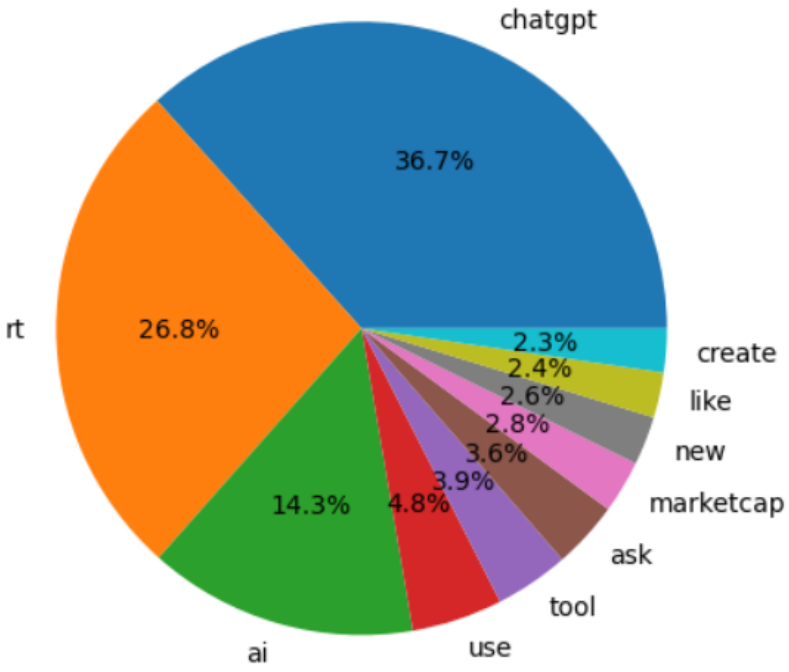
Figure 13: Like count vs Retweet scatter plot

The scatter plot illustrates the relationship between likes and retweets for tweets that received responses.

- Points close to a straight line indicate a strong positive correlation, suggesting that more likes correspond to more retweets.
- Distant or clustered points suggest a weak or no correlation between likes and retweets.
- The scatter plot shows that several tweets garnered both high likes and retweets, indicating they are popular content.
- Many tweets received less than 100 likes and retweets, indicating lower engagement levels.
- Some tweets have high likes but few retweets, or vice versa, possibly due to various factors such as tweet content or hashtags used.

17. TOPIC MODELLING (HIGHLIGHTING KEYWORDS)

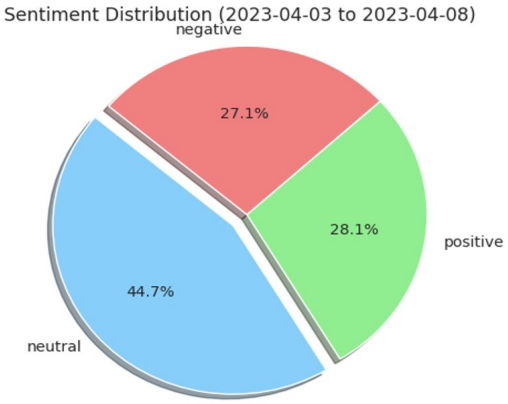
To highlight keywords in the text column, we can use the NLP (Natural Language Processing) method, a frequently used tool for analyzing text data. It allows you to process and analyze the text, breaking it into individual words, defining their parts of speech, highlighting nouns, verbs, etc. Then we can analyze the information received and highlight the most common words in the text.

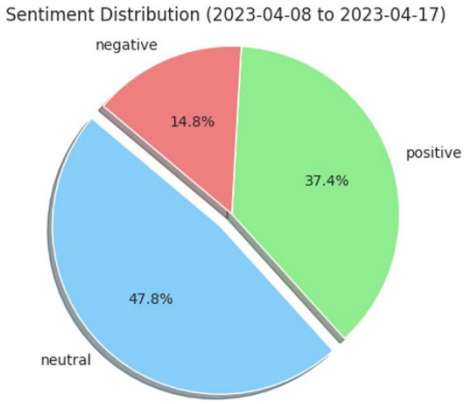


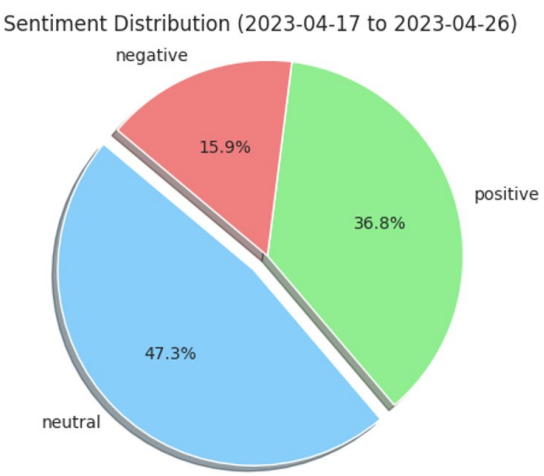
The fact that "chatgpt" is the most often used keyword suggests that the data was likely gathered using the ChatGPT model. "RT" is the second most popular keyword, which may be because many Twitter users use it to refer to retweets. The third most common usage of the phrase "ai" shows how important artificial intelligence technologies are to the field of data analysis. Other widely used keywords include "use," "tool," "ask," "marketcap," "new," and "like," which may suggest their importance in the context of data analysis.

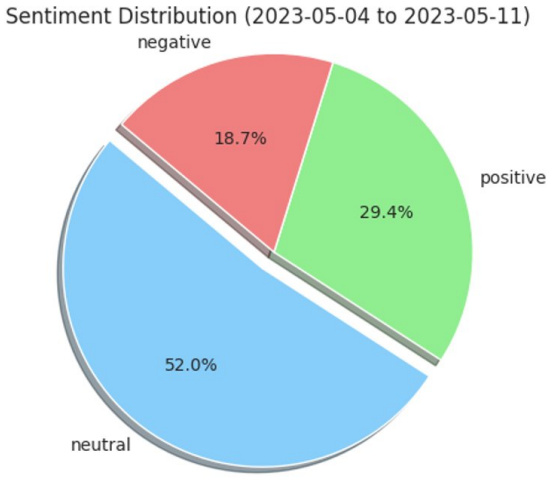
18. SENTIMENT ANALYSIS:

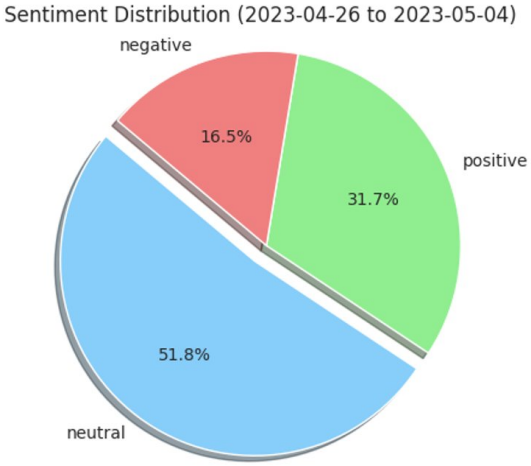
1) Sentiment Analysis based on different time period











Positive Sentiment: The data shows the percentage of positive sentiment for different date ranges. The highest positive sentiment is observed during the second date range (2023-04-08 to 2023-04-17) with approximately 37.37%. This indicates that during this period, there was a relatively higher level of positive sentiment expressed in the data compared to other date ranges. Positive sentiment may reflect positive experiences, satisfaction, or approval expressed by individuals in their posts, comments, or reviews.

Neutral Sentiment: The data also includes the percentage of neutral sentiment for each date range. Neutral sentiment indicates a lack of strong positive or negative emotions in the expressed opinions. The highest neutral sentiment is observed during the fifth date range (2023-05-04 to 2023-05-11) with approximately 51.97%. This suggests that during this period, a significant proportion of sentiments expressed were neither overwhelmingly positive nor negative.

Negative Sentiment: The data provides the percentage of negative sentiment for each date range. Negative sentiment reflects dissatisfaction, criticism, or negative opinions expressed in the data. The highest negative sentiment is observed during the fifth date range (2023-05-04 to 2023-05-11) with approximately 18.68%. This indicates that during this period, there was a relatively higher level of negative sentiment expressed compared to other date ranges.

Overall Trends: The data shows that sentiment percentages are not constant and can vary over different date ranges. This suggests that external factors, events, or changes in the environment might influence the sentiments expressed by individuals.

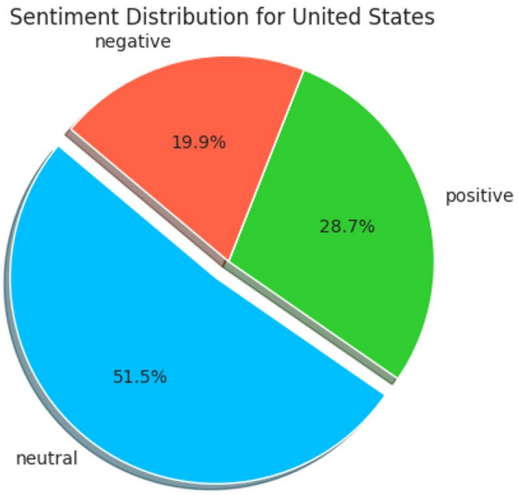
2)Sentiment Analysis by locations

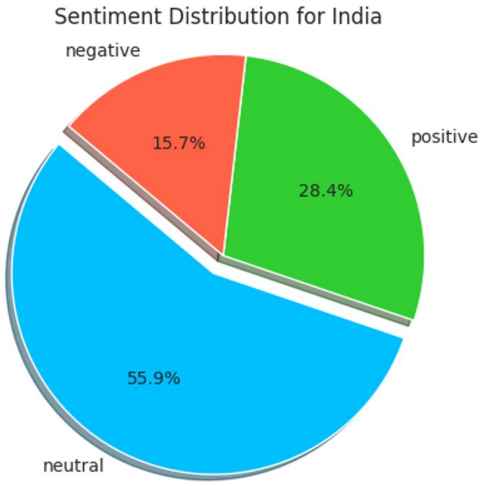
```
1 unique_locations_sorted = data['user_location'].value_counts().sort_values(ascending=False)
2 print(unique_locations_sorted)
```

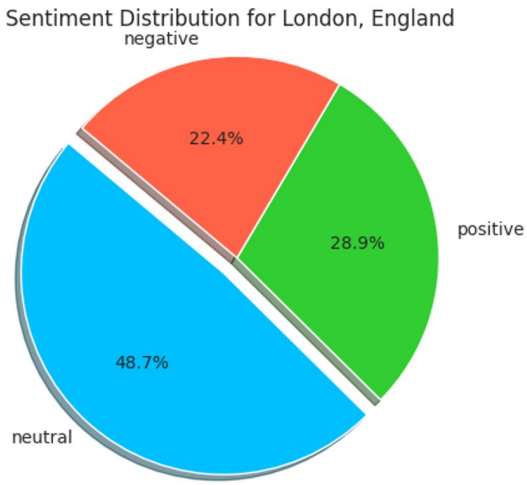
```
United States                  136
India                          102
London, England                 76
Lagos, Nigeria                  76
日本                               75
                               ...
徳島県                               1
with Jermaine Lamar Cole.       1
Mayur Vihar Ph 1 Extn, Delhi    1
Cirebon, Jawa Barat             1
London UK                       1
Name: user_location, Length: 6634, dtype: int64
```

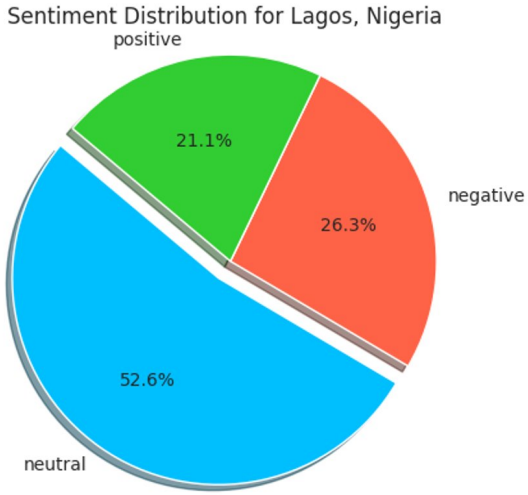
Figure 14: Sentiment analysis by locations

1. **United States**: This location has the highest count of sentiments, with 136 occurrences in the dataset. The sentiment analysis can provide valuable insights into the overall sentiment trends of users or content originating from the United States.
2. **India:** India is the second-most prevalent location in the dataset, with 102 occurrences. Analyzing sentiments from India can offer important insights into the sentiments expressed by users or content related to this country.
3. **London, England:** This specific location has 76 occurrences in the data. Sentiment analysis for London can be useful for understanding the sentiment trends associated with this particular city.
4. **Lagos, Nigeria:** Lagos also has 76 occurrences in the data. Analyzing sentiments from Lagos, Nigeria, can provide insights into the sentiment patterns related to this location.









The data shows the percentage of positive, neutral and negative user sentiment for 4 different locations:

- United States: Users have a slightly higher positive sentiment at 28.67% compared to the other locations. The majority of users, at 51.47%, have a neutral sentiment.
- India: Users have a similar positive sentiment of 28.43% as the US. The majority at 55.88% have a neutral sentiment.
- London, England: Users have the highest negative sentiment at 22.37%. While the positive sentiment is close to the US and India at 28.95%, the neutral sentiment is the lowest at 48.68%.
- Lagos, Nigeria: Users have the lowest positive sentiment at 21.05% and the highest negative sentiment at 26.32%. The majority at 52.63% have a neutral sentiment.

In summary, users in the US, India and London seem to have a similar distribution of sentiment, with a majority of neutral users and around 28-29% positive users. Users in Lagos, Nigeria has a lower positive sentiment and higher negative sentiment compared to the other locations.

19. LABELLING THE DATASET

- Our next step was to convert unsupervised learning to supervised learning problem. For that we need to label the dataset and include target variable as positive and negative sentiment (we are not considering neutral tweets as it will become more complex and training the model would take time). Moreover, neutral case will be considered in future when deploying the model through AWS as it can handle the complexity of model.

- Textblob is used for labelling the dataset with positive and negative sentiment and complete code can be seen in below figure.

```
max_len = 100
max_words = 10000

tokenizer = Tokenizer(num_words=max_words, lower=True)
tokenizer.fit_on_texts(df['cleaned_text'])
sequences = tokenizer.texts_to_sequences(df['cleaned_text'])
word_index = tokenizer.word_index
data = pad_sequences(sequences, maxlen=max_len)
```

```
from textblob import TextBlob
def get_sentiment_polarity(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity
```

```
df['sentiment_polarity'] = df['cleaned_text'].apply(get_sentiment_polarity)
df['sentiment'] = np.where(df['sentiment_polarity'] >= 0, 'positive', 'negative')
```

```
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(df['sentiment'])
labels = labels.reshape(-1, 1)
```

20. TRAINING AND BUILDING NEURAL NETWORK:

A) Training the simple neural network model: Data set was split with 80% training and 20% validation to train simple neural network.

```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
```

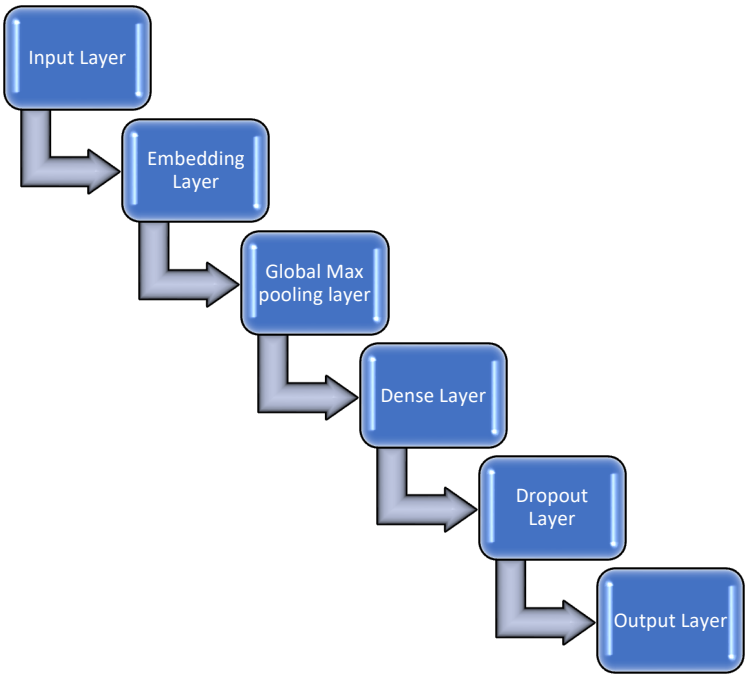


Fig. neural network layer

- Training of the neural network was done with 6 layers.
    1. Input Layer: This line creates an input layer that specifies the shape of the input data. max_len is the maximum length of the input sequences. This input layer will be used to feed sequences of tokens (words or characters) into the model.
    2. Embedding Layer: The Embedding layer is used to convert the input sequences of integer indices (representing words) into dense vectors of fixed size. It's often used in natural language processing (NLP) tasks to learn word representations. In this case, the layer will create word embeddings of 128 dimensions for each word in the input sequence. len(word_index) + 1 represents the vocabulary size (number of unique words) plus one for out-of-vocabulary words. input_length=max_len specifies the length of each input sequence.
    3. Global Max Pooling Layer: The GlobalMaxPooling1D layer takes the maximum value across the entire sequence of embeddings. It reduces the dimensionality of the output from the embedding layer to a fixed-size vector by selecting the most important features for each dimension across the sequence. This allows the model to focus on the most relevant information while keeping the overall structure of the sequence intact.
    4. Dense Layer: This line adds a dense (fully connected) layer with 64 units and a ReLU activation function to the model. The output of the global max pooling layer is connected to this dense layer, allowing the model to learn higher-level features and patterns from the pooled embeddings.
    5. Dropout Layer: The Dropout layer is used for regularization to prevent overfitting. It randomly sets a fraction of the input units to 0 during training (in this case, 50% of the units). This helps in generalization and improves the model's ability to handle unseen data.

6. Output Layer: The output layer consists of a single neuron with a sigmoid activation function. This is used for binary classification tasks, where the model outputs a probability between 0 and 1, representing the likelihood of a positive class (1).

```
input_layer = Input(shape=(max_len,))
embedding_layer = Embedding(len(word_index) + 1, 128, input_length=max_len)(input_layer)
x = GlobalMaxPooling1D()(embedding_layer)
x = Dense(64, activation='relu')(x)
x = Dropout(0.5)(x)
output_layer = Dense(1, activation='sigmoid')(x)

model = Model(inputs=input_layer, outputs=output_layer)
```

- Compile and train the model:

1. The model is compiled using the Adam optimizer, binary cross-entropy loss (since it's a binary classification problem), and accuracy as the evaluation metric. Early stopping and model checkpoint callbacks are set up to stop training when the validation loss does not improve for three consecutive epochs and to save the best model based on the validation loss.

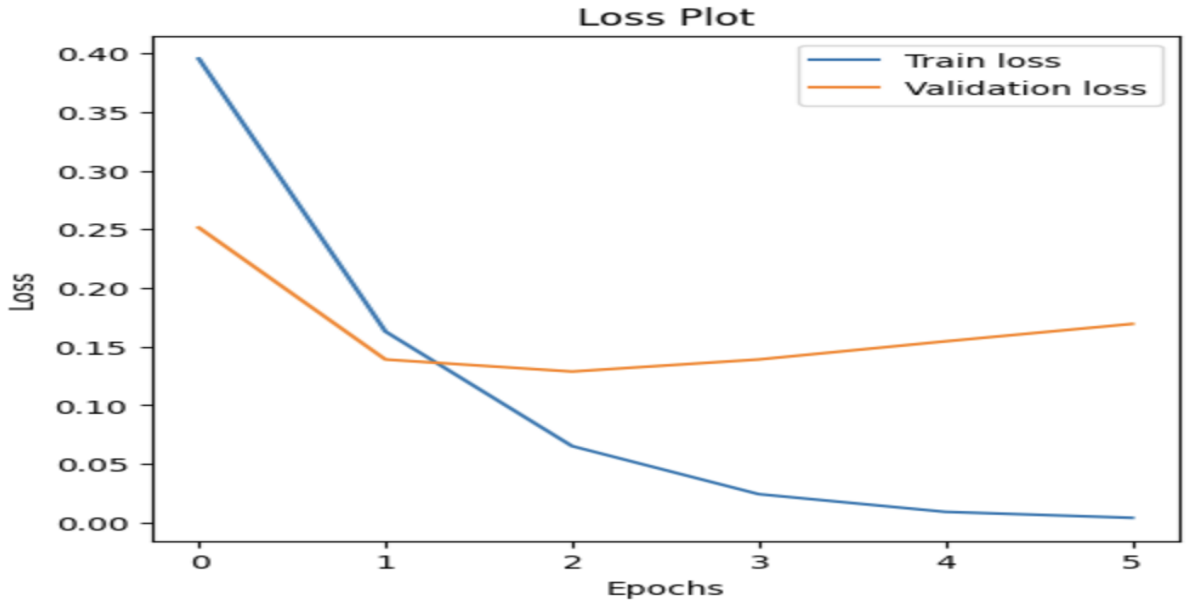
2. The model is then trained on the training data ('X_train' and 'y_train') for 10 epochs, using a batch size of 32. The validation data ('X_test' and 'y_test') is used for validation during training. The training history is stored in the 'history' variable.

```
model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

callbacks = [EarlyStopping(monitor='val_loss', patience=3),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]

history = model.fit(X_train, y_train, batch_size=32, epochs=10, validation_data=(X_test, y_test), callbacks=callbacks)
```

21. EVALUATING MODEL PERFORMANCE:

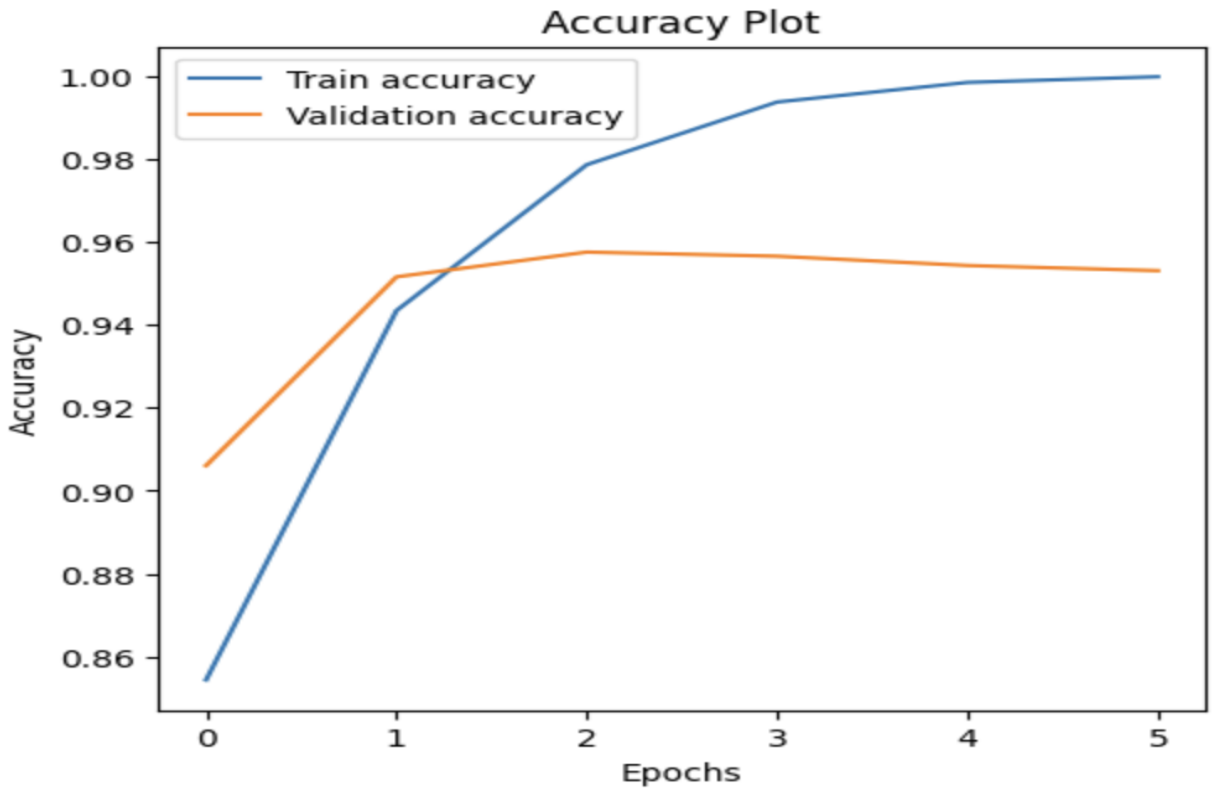
- Learning curves are plots that show changes in learning performance over time in terms of experience.
- Learning curves of model performance on the train and validation datasets can be used to diagnose an underfit, overfit, or well-fit model.
- Learning curves of model performance can be used to diagnose whether the train or validation datasets are not relatively representative of the problem domain.
- **Epoch**: Each time a dataset passes through an algorithm, it is said to have completed an epoch. Therefore, Epoch, in machine learning, refers to the one entire passing of training data through the algorithm. It's a hyperparameter that determines the process of training the machine learning model.
- **Batch:** The training data is always broken down into small batches to overcome the issue that could arise due to storage space limitations of a computer system. These smaller batches can be easily fed into the machine learning model to train it. This process of breaking it down to smaller bits is called batch in machine learning. This procedure is known as an epoch when all the batches are fed into the model to train at once.

- **A.) Optimization Learning Curves (Loss Plot)**: Learning curves calculated on the metric by which the parameters of the model are being optimized. the loss curve, or training loss curve, gives us insights into how the model's performance improves over time by measuring the error or dissimilarity between its predicted output and the true output. The loss represents how far off the model's predictions are from the actual values. By minimizing the loss, the model aims to make its predictions as close as possible to the true values. So, the loss curve shows us how the model's error decreases as it learns, which indicates an improvement in its performance. Based on loss plot we can deduce whether algorithm is overfitting, underfitting or good fit. Overfitting is defined as good performance on the training data, poor generalization to other data. Underfitting is defines as poor performance on the training data and poor generalization to other data. From fig below we can see that as number of epochs is increasing the gap between train loss and validation is growing which suggests there might be overfitting issue if we increase the epoch further. Thus, best epoch in our case should be 2 where both graphs intersect.



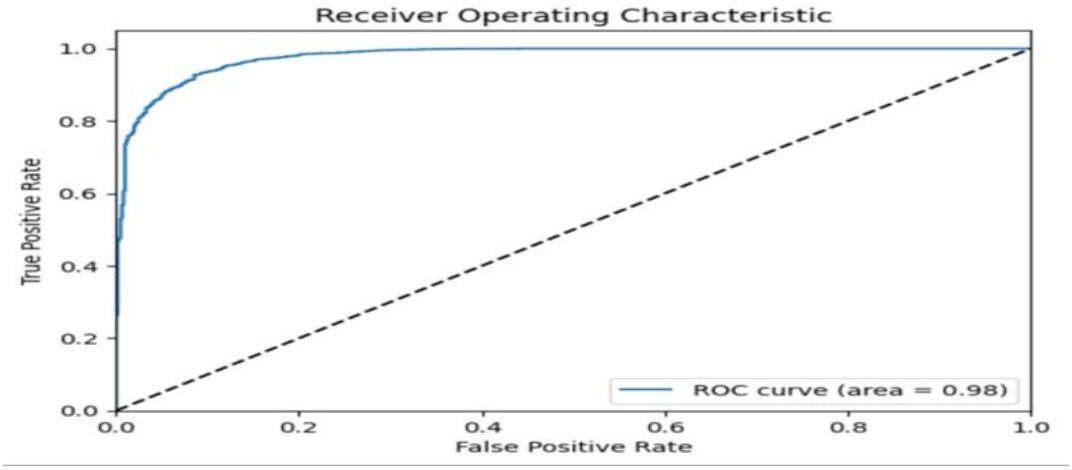
B) **Performance Learning Curves (Accuracy Curve)**: Learning curves calculated on the metric by which the model will be evaluated and selected. The accuracy curve, also known as the training accuracy curve, shows us how good the model is at making correct predictions on the training data as it goes through the training process. Accuracy is measured in percentages and tells us the proportion of instances the model correctly classified out of the total number of instances. So, the accuracy curve gives us a sense of how well the model fits the training data and improves its ability to make accurate predictions.

Accuracy plot suggests that with increase in epoch accuracy increases for both training and validation, but gap between those widen which indicates us to stop after epoch=2.



C) **Receiver Operator Characteristic (ROC)**: A ROC curve is constructed by plotting the true positive rate (TPR) against the false positive rate (FPR). The true positive rate is the proportion of observations that were correctly predicted to be positive out of all positive observations (TP/(TP + FN)). Similarly, the false positive rate is the proportion of observations that are incorrectly predicted to be positive out of all negative observations (FP/(TN + FP)). ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR). Classifiers that give curves closer to the top-left corner indicate a better performance.

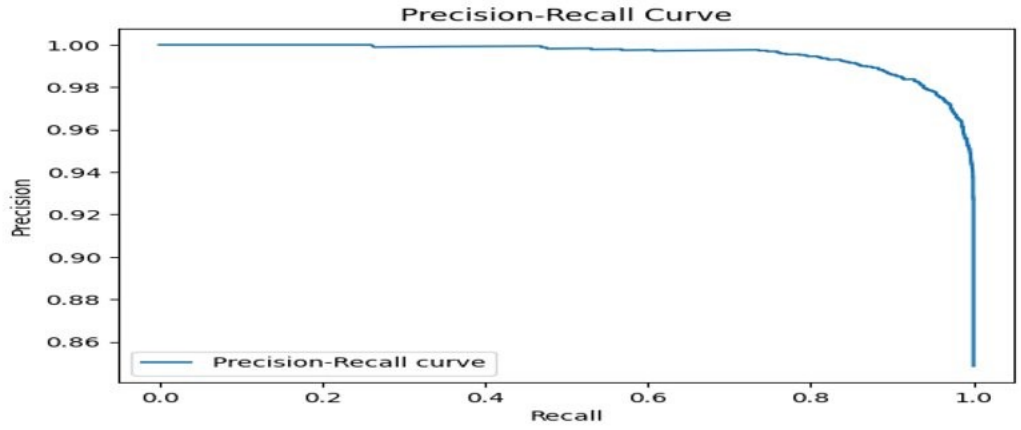
ROC curve for the neural network lies in upper left corner which indicates model is performing good. Moreover, area under curve (AUC+ is equals to 0.98, which further supports the analysis.



D) **Precision and recall curve:** Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

The precision-recall curve shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning most of all positive results (high recall).

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly.



22. FUTURE IMPLEMENTATION AND DEPLOYMENT IN AWS CLOUD:

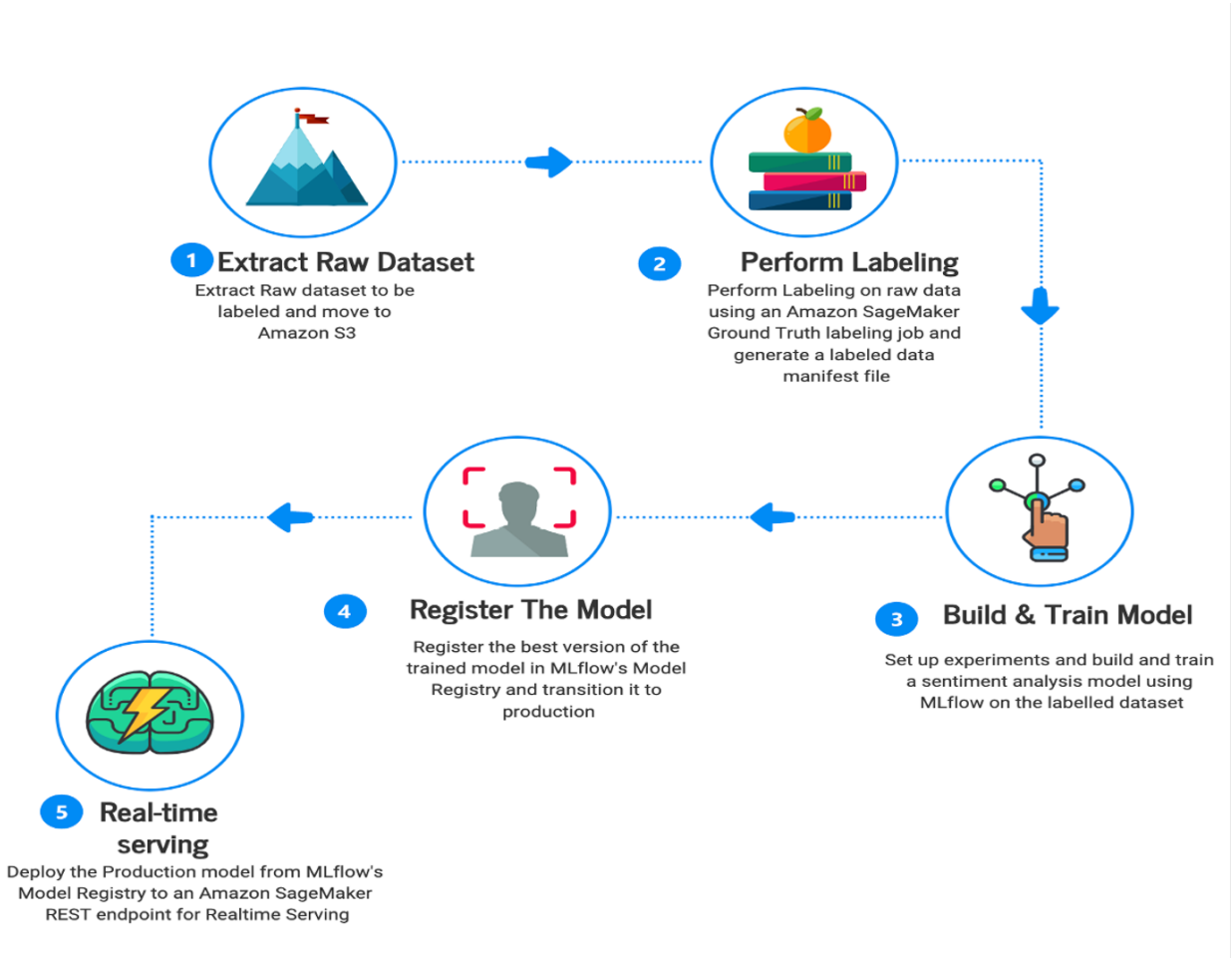


Figure: process flow diagram to deploy model in AWS

I. Extract raw dataset and building ETL pipeline:

AWS Glue is another offering from AWS and is a serverless ETL (Extract, Transform, and Load) service on the cloud. It is fully managed, cost-effective service to categorize your data, clean and enrich it and finally move it from source systems to target systems.

A.) Key Feature of AWS Glue:

1. AWS Glue automatically generates the code structure to perform ETL after configuring the job.
2. You can modify the code and add extra features/transformations that you want to carry out on the data.
3. AWS crawler, connect to data sources, and it automatically maps the schema and stores them in a table and catalog.

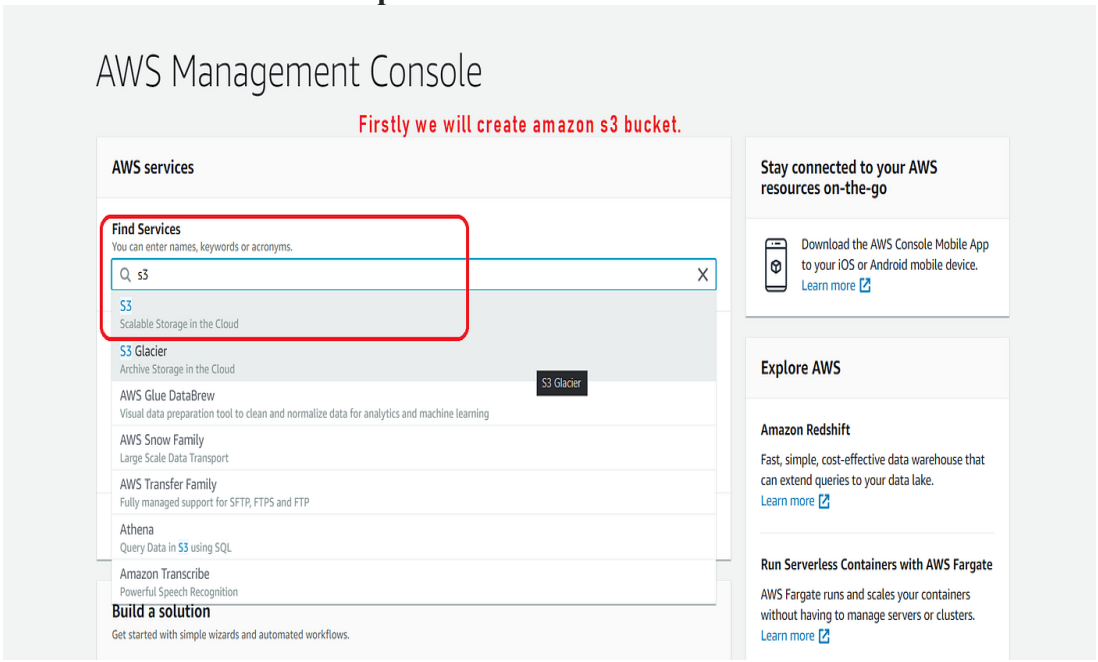
B.) Main Steps to create ETL pipeline in AWS Glue:

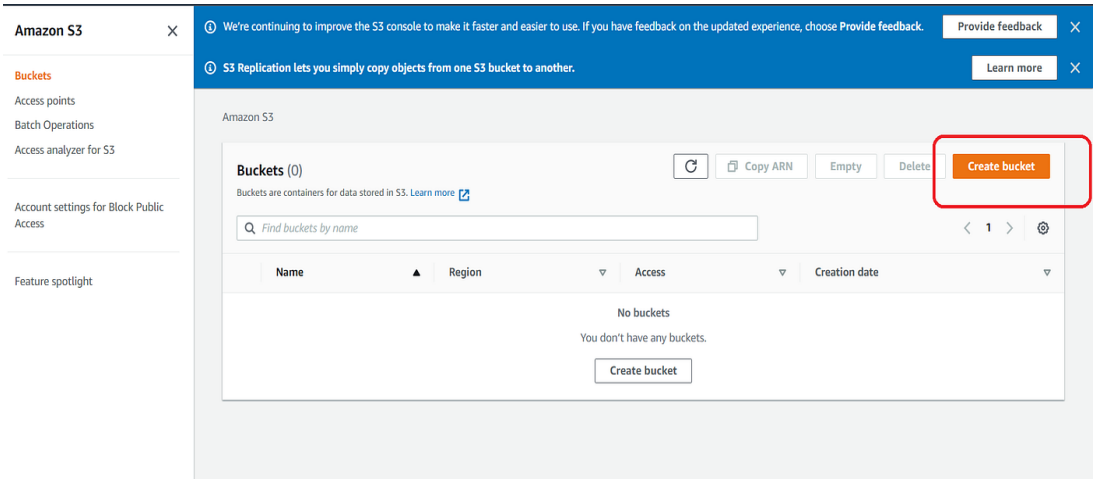
I. Create a Crawler
II. View the Table

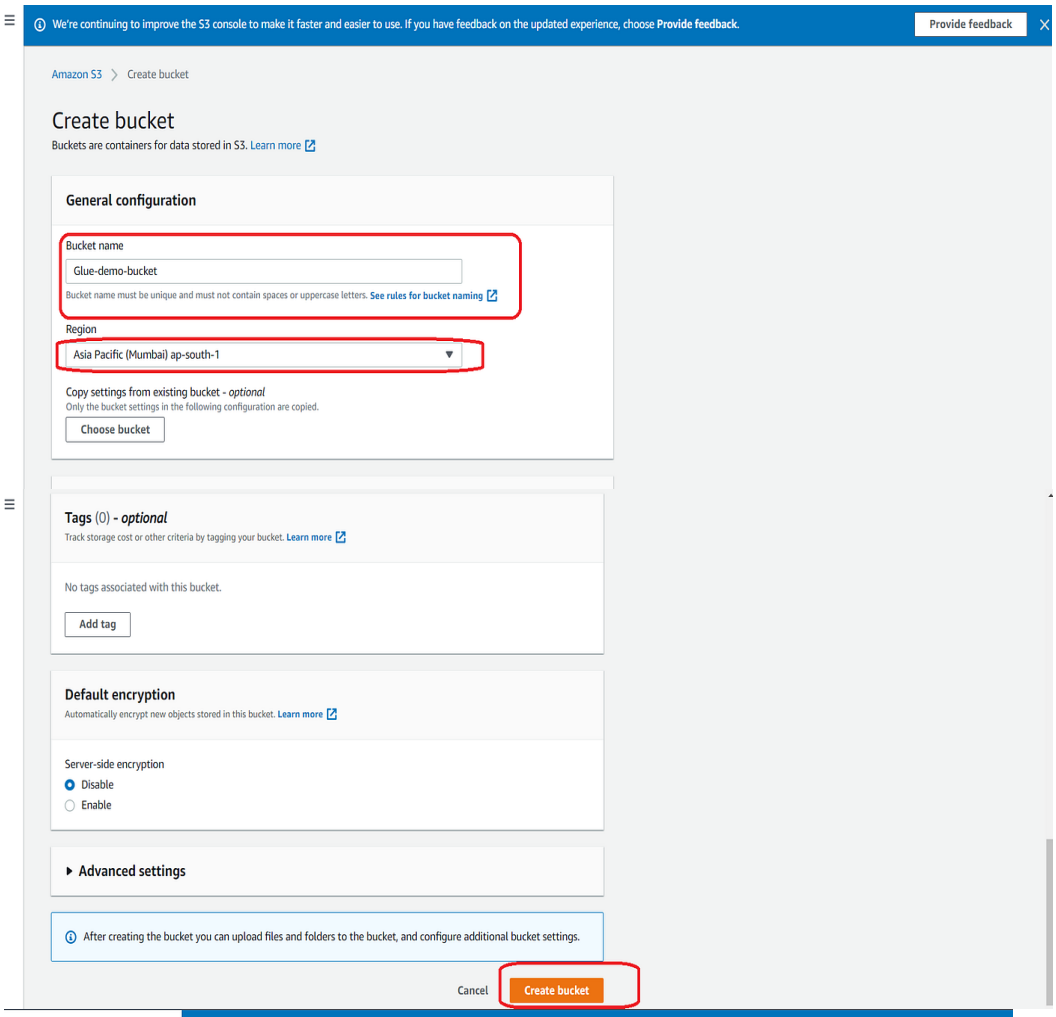
III. Configure Job

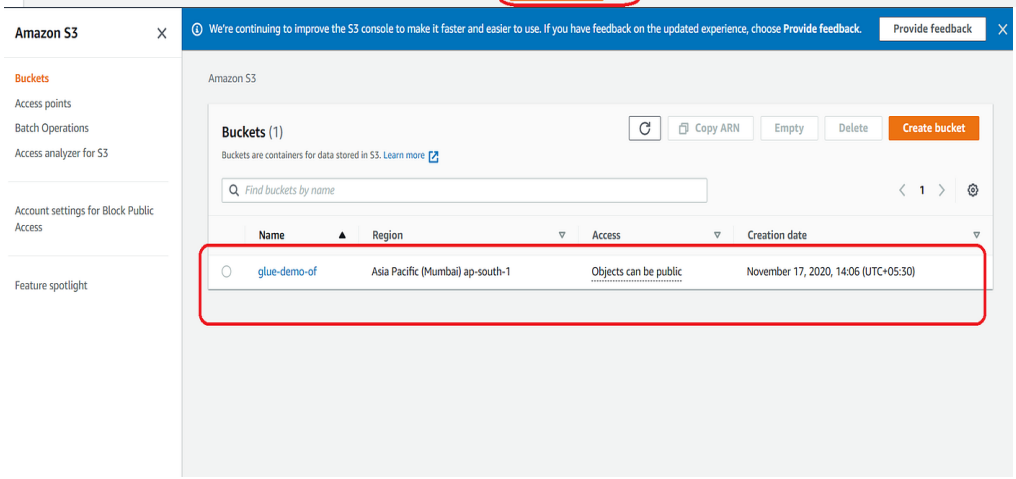
C.) STEP BY STEP GUIDE ON HOW TO BUILD ETL PIPELINE

1. Create the S3 Bucket and upload the data source csv file in S3 Bucket.



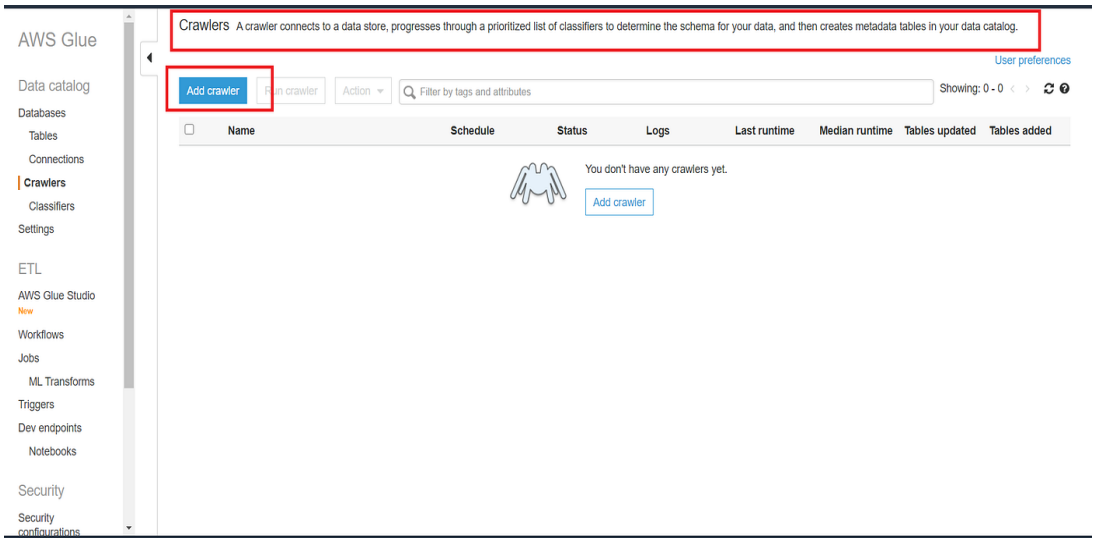




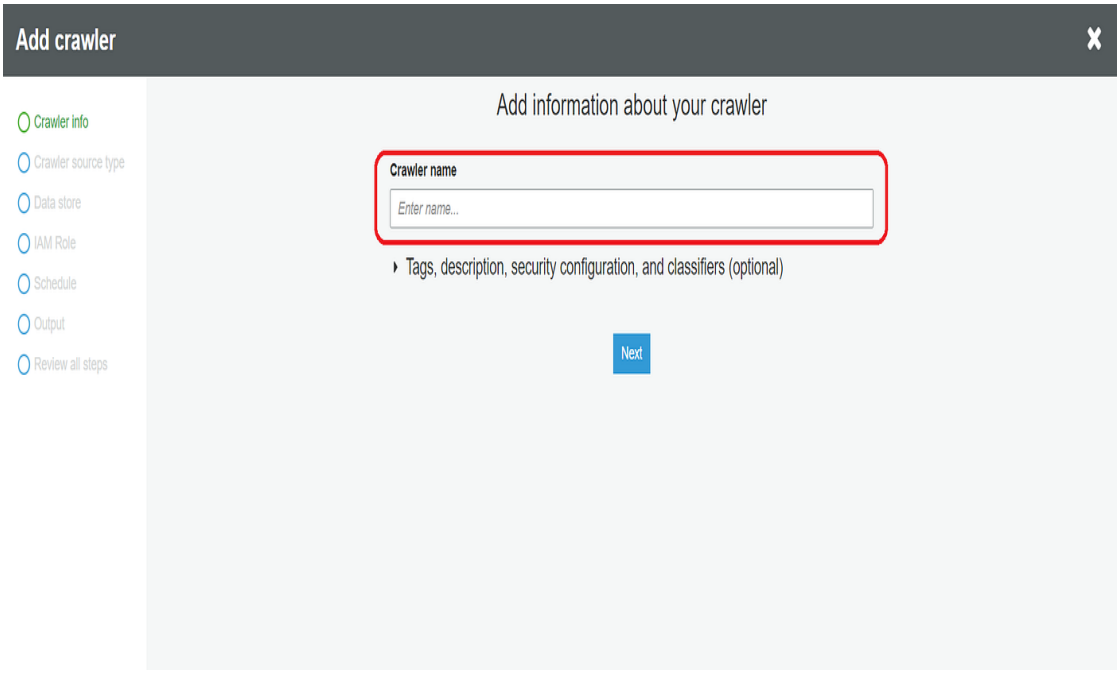


2.) Create a Crawler

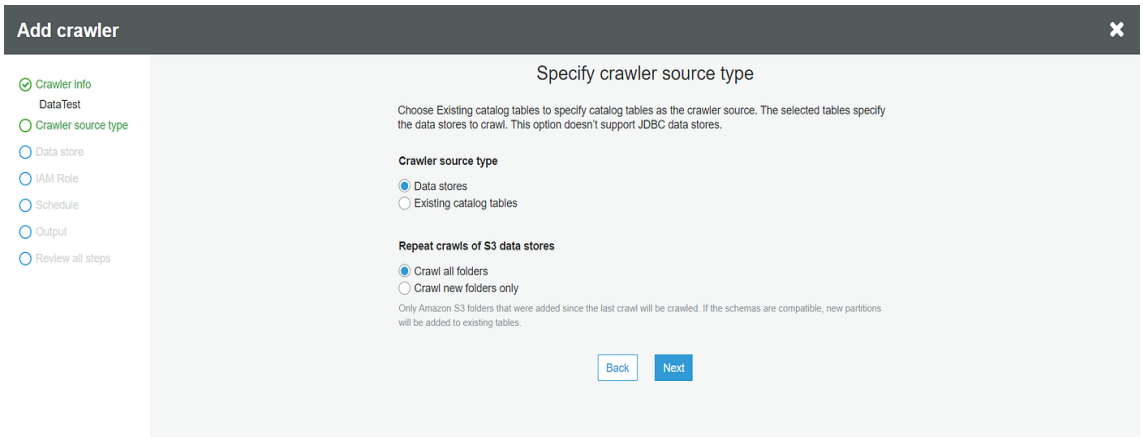
1. Sign in to AWS Console, and from the search option, search AWS Glue and click to open AWS Glue page.
2. Go to console Add crawler.



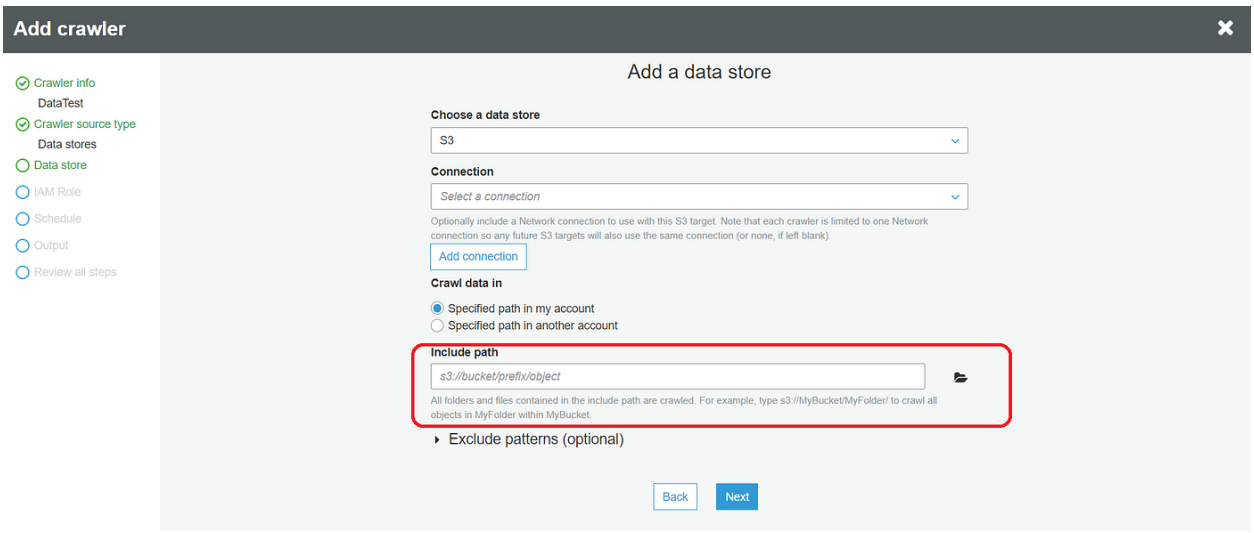
3. Once you click Add Crawler then specify the crawler name.



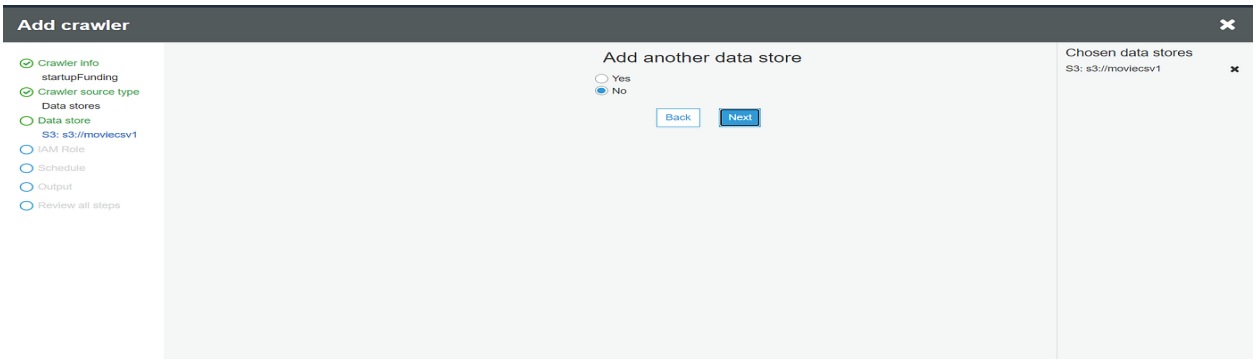
4. After specifying the name, click **Next** and on the next screen, select the data source and click **Next**.



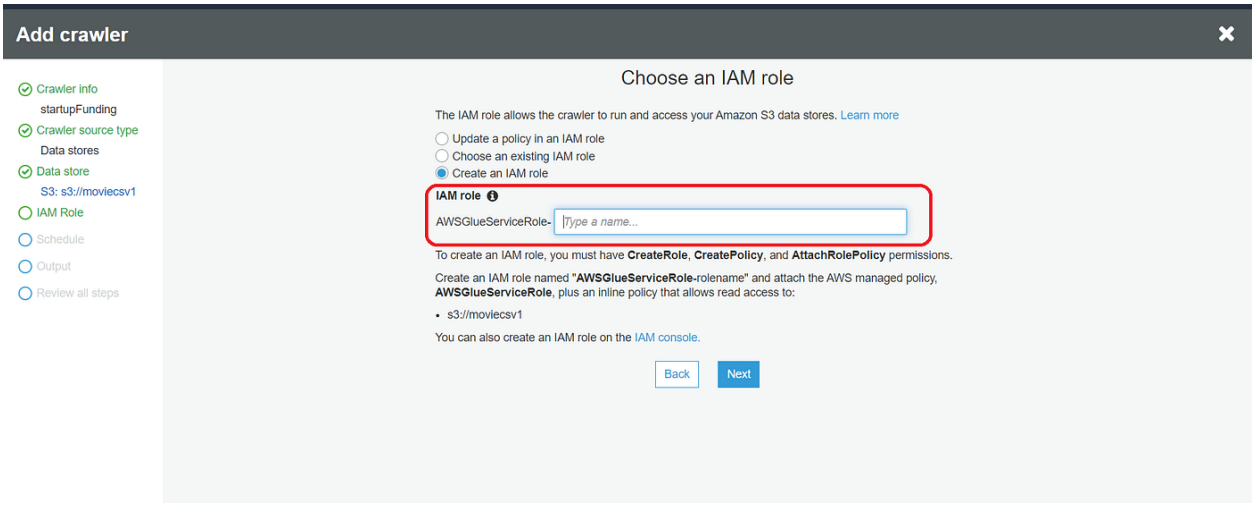
5. On the next screen, select the Data Source as S3, and specify the path of the data.



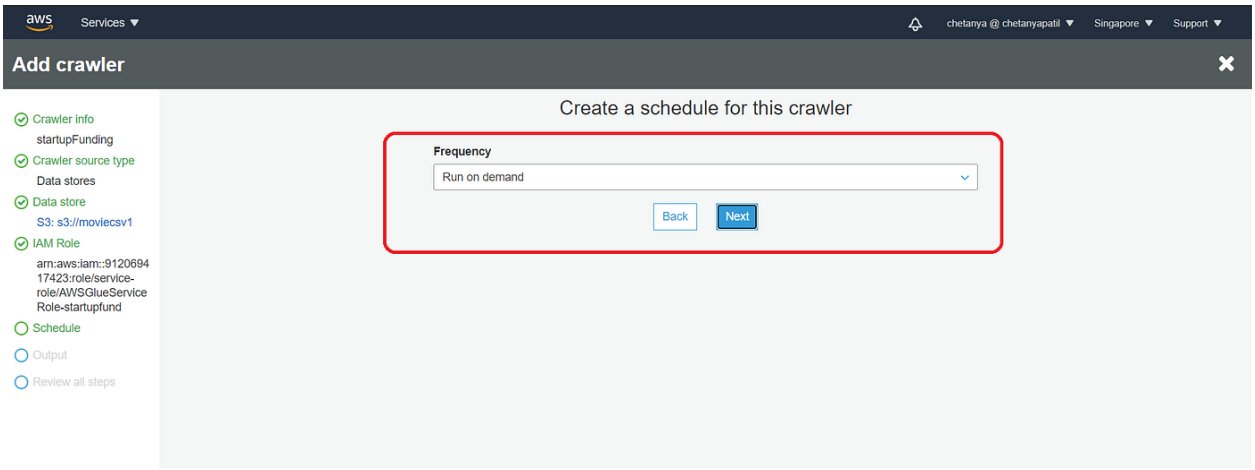
6. Once you fill all the information, click on Next, and in the next section, select No when asked for Add another data source.



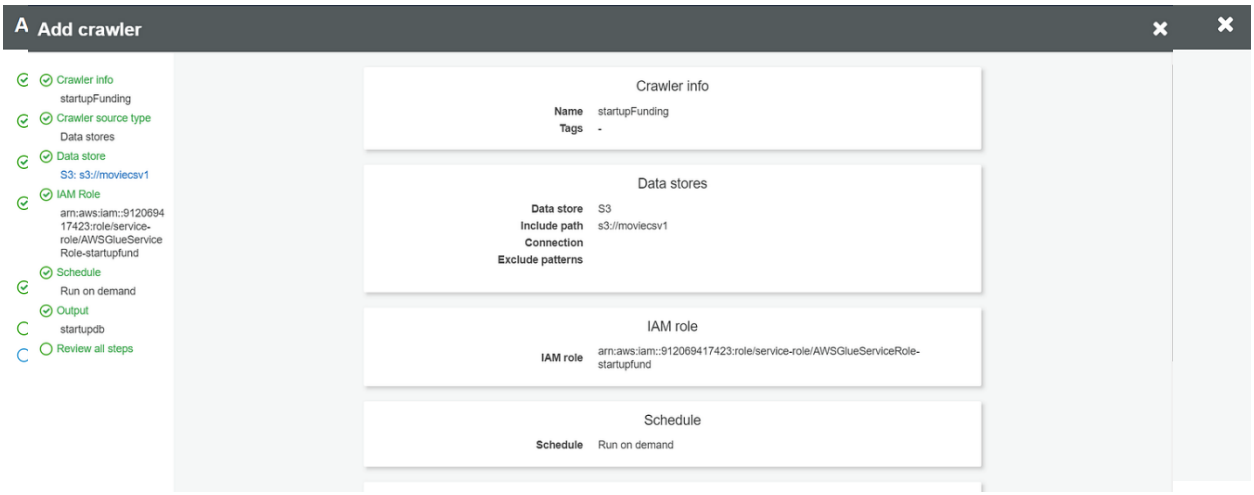
7. Once you click Next, it will ask you to create an IAM role to access S3 and run the job. Provide the name of the role and click Next.

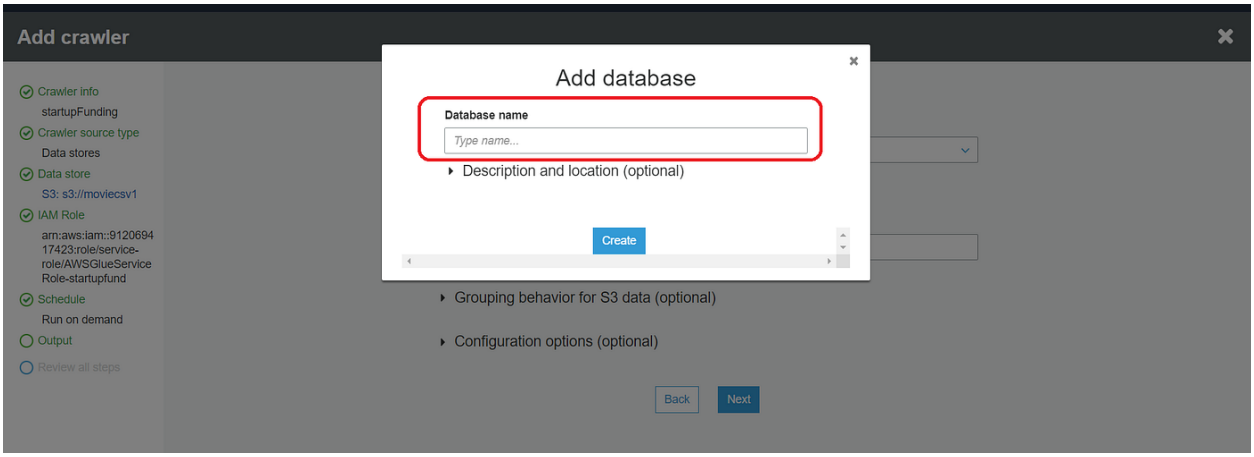


8. Once you provide the IAM role, it will ask how you want to schedule your crawler.

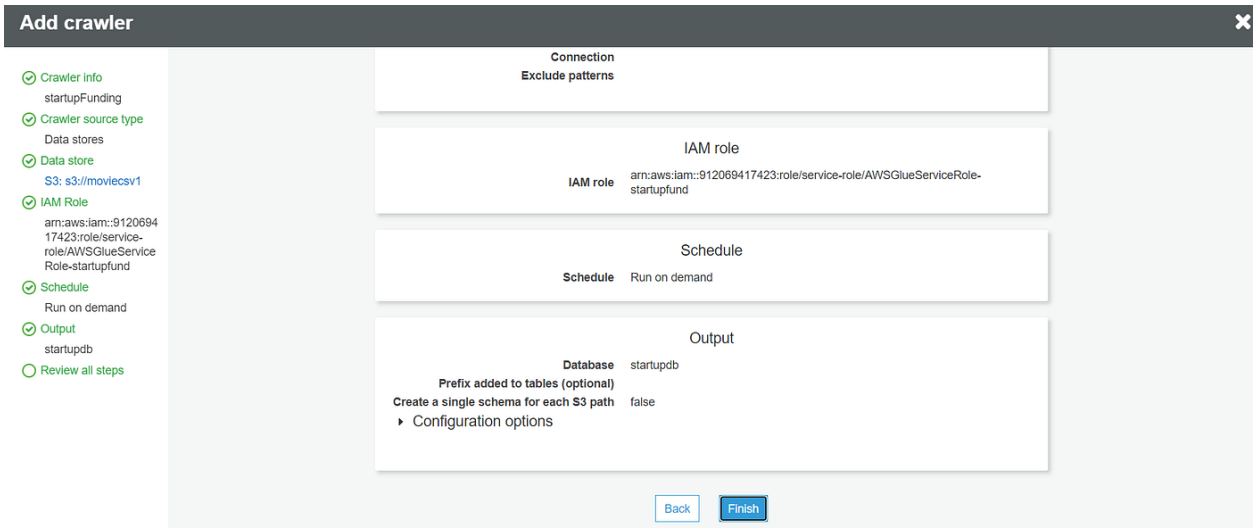


9. Once you select the scheduler, click next and create the output database.

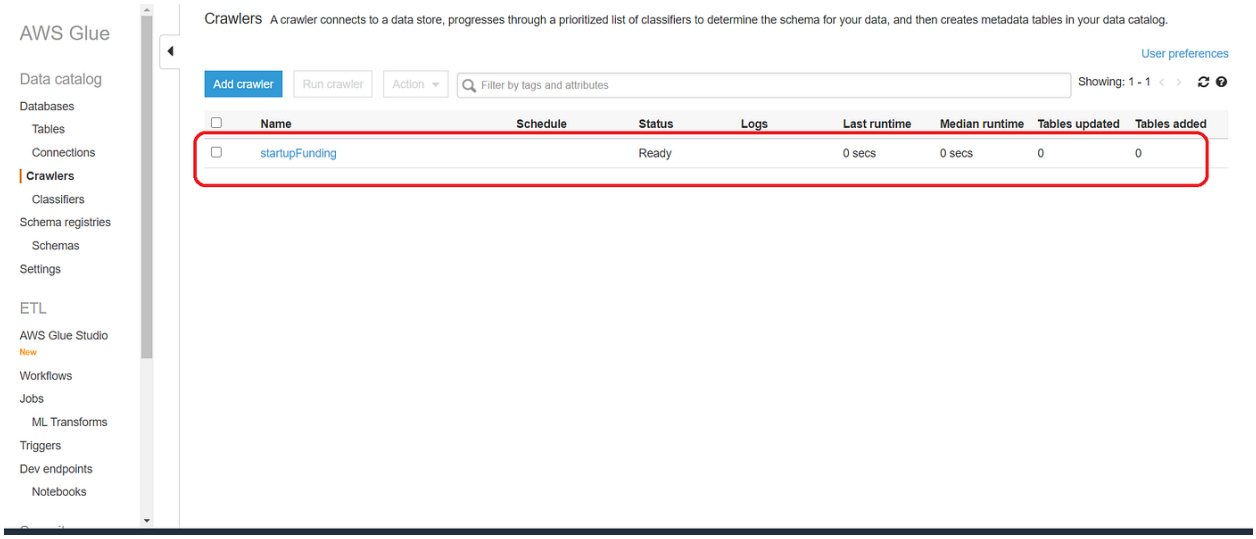




10. Once you create the database, the review page will be open. Review all the settings and click finish.



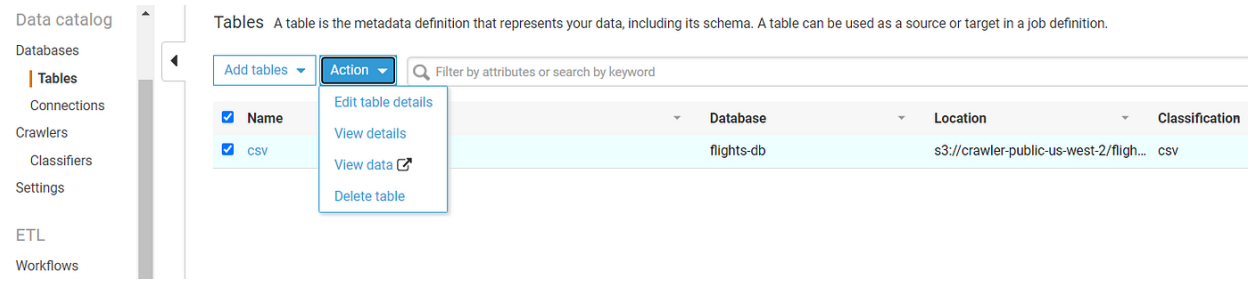
11. Once you click on finish, the crawler will be created, instantly, and it is available for run. Click on Run Crawler, to start execution.



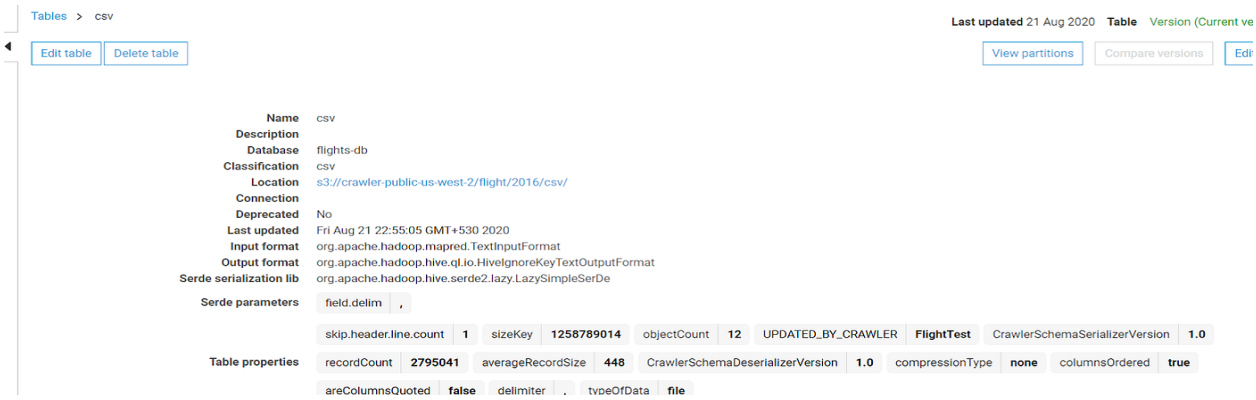
3.) View the Table

Once the crawler is successfully executed, you can see the table and its metadata created in the defined DB. Steps to explore the created table —

1. On the bottom section of Glue Tutorial, click on Explore table.
2. This will head you to the tables section and click on the CSV table created inside flights-db database.
3. Click on the table and click View Details.



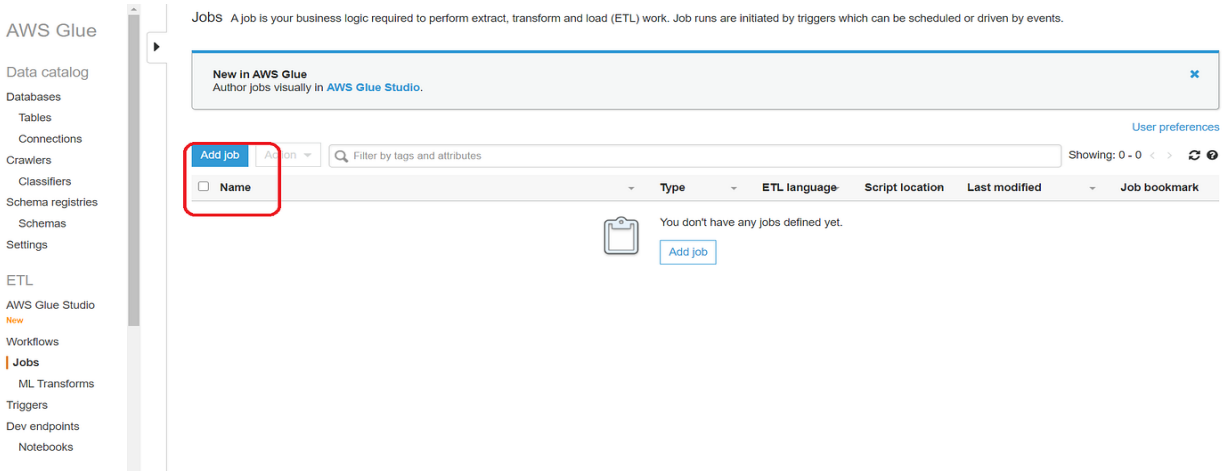
4. You can see all the information/properties of the table.



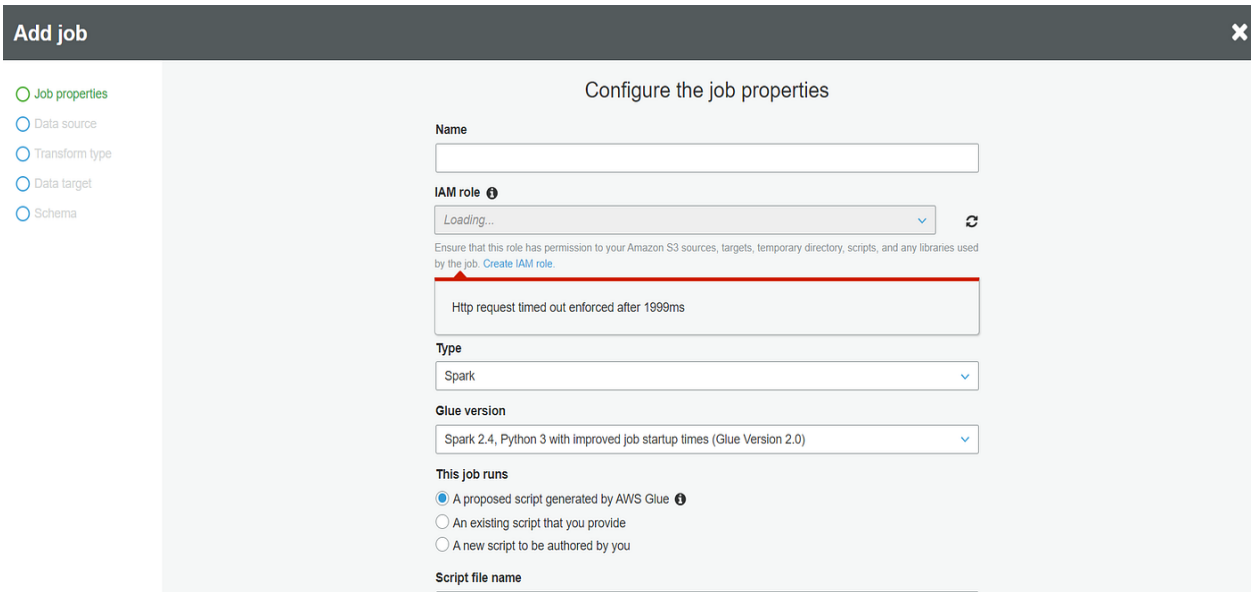
5. If you scroll down further, to the same page, you can see the metadata of the table fetched automatically from the file.

4.) Configure Job

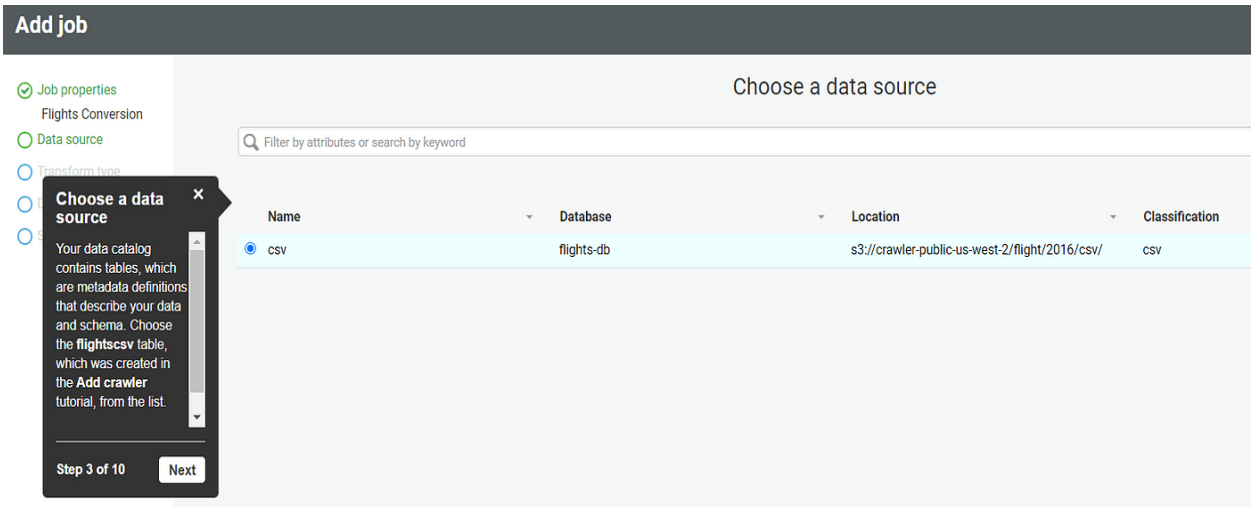
1. In the section, we have to configure the job to move data from S3 to the table by using the crawler.



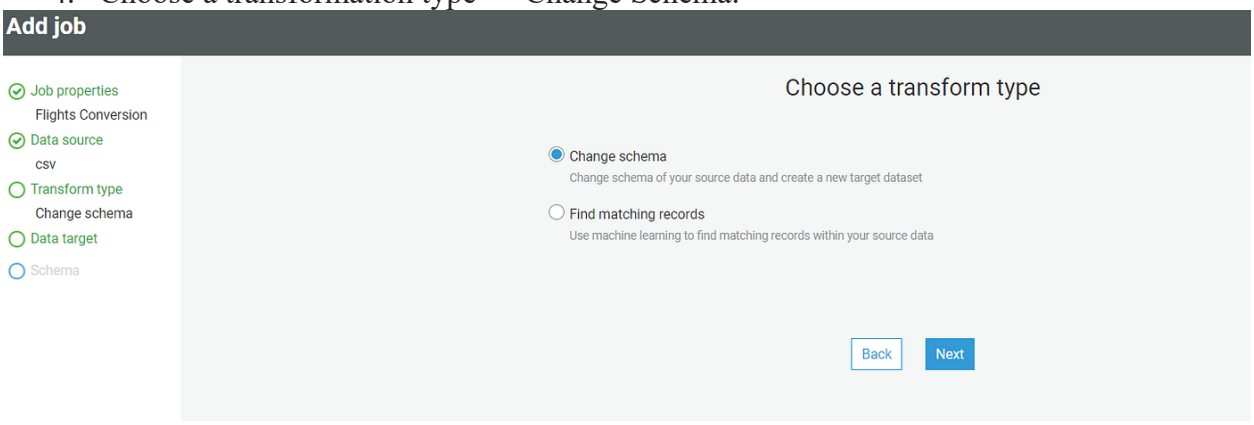
2. Once you click on Add job, a job configuration page will open up. Fill the required details like name of the job, select the IAM role, type of execution and other parameters.



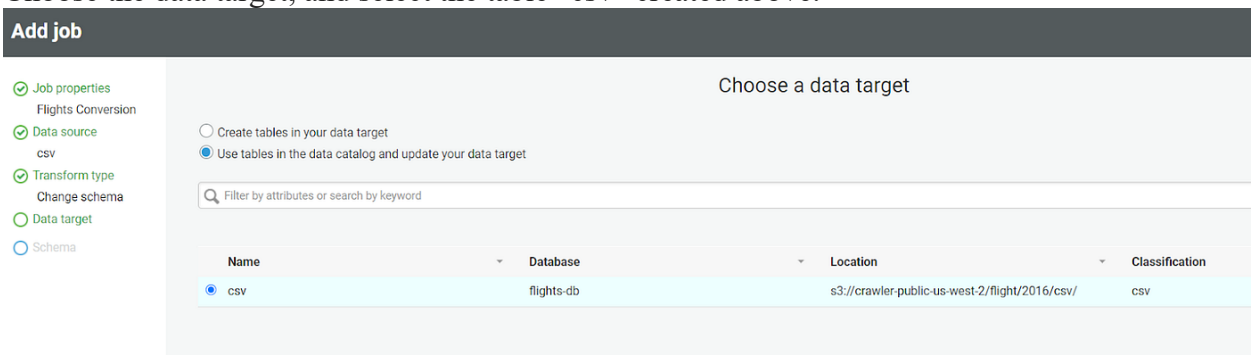
3. After the configuration click on Next. On the next page select the data source "csv" and click on next.



4. Choose a transformation type — Change Schema.

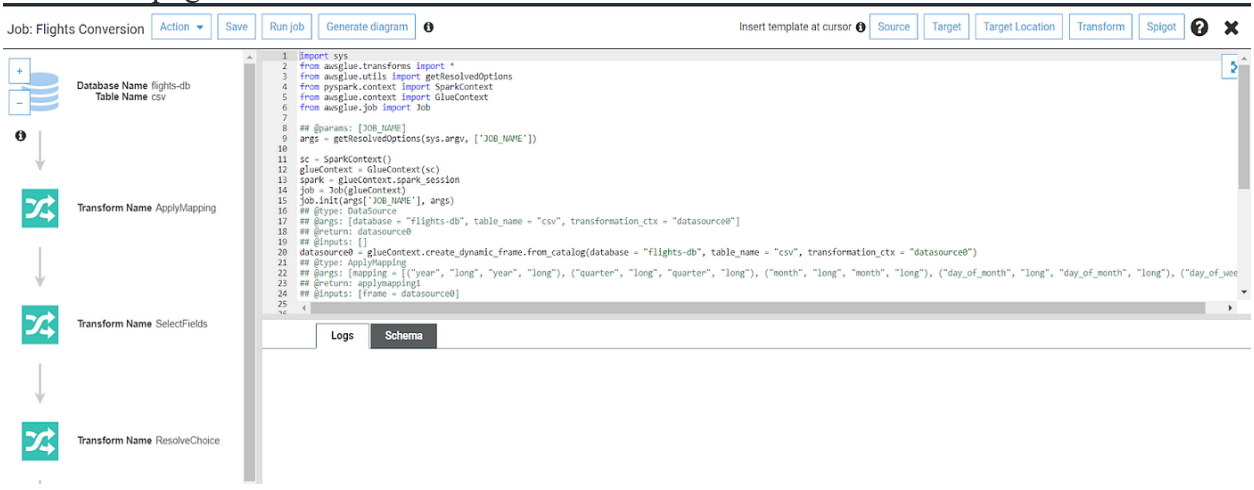


5. Choose the data target, and select the table "csv" created above.

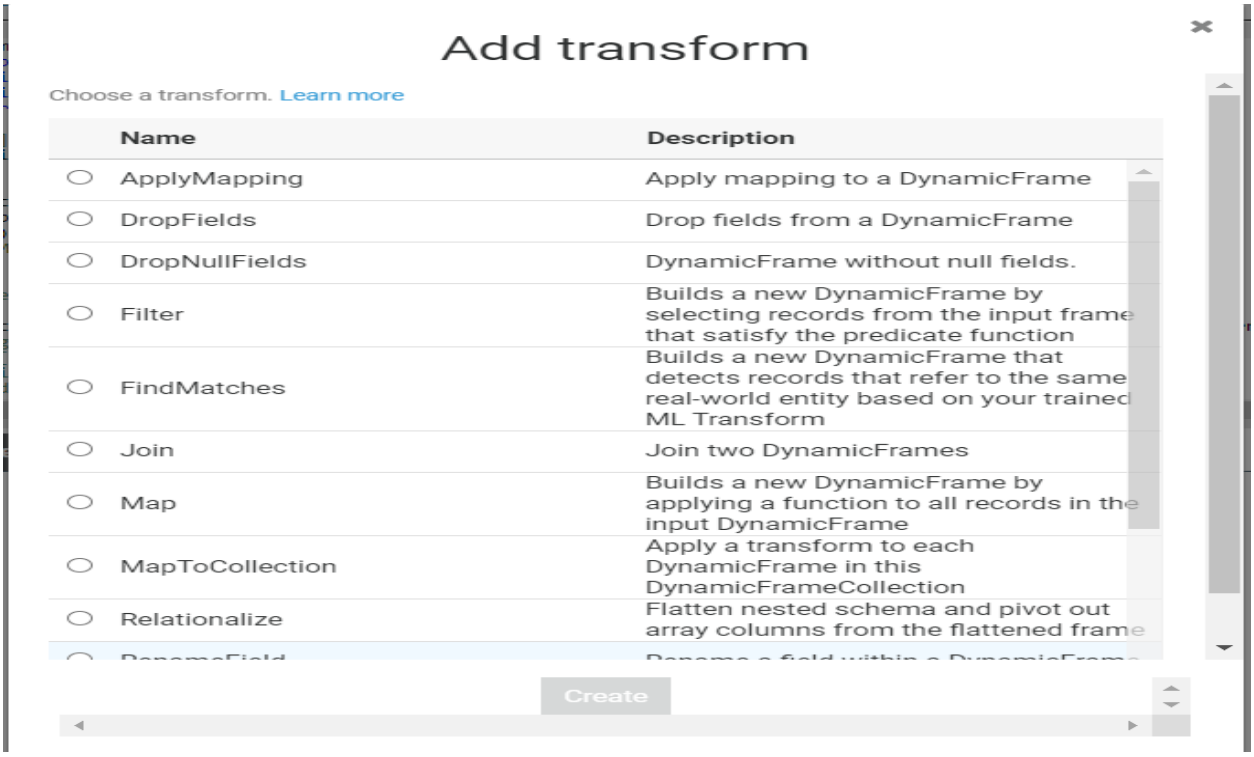


6. On the next page, you'll see the source to the target mapping information. Add/delete the columns that you wish to. We suggest keeping the default mapping and click next.

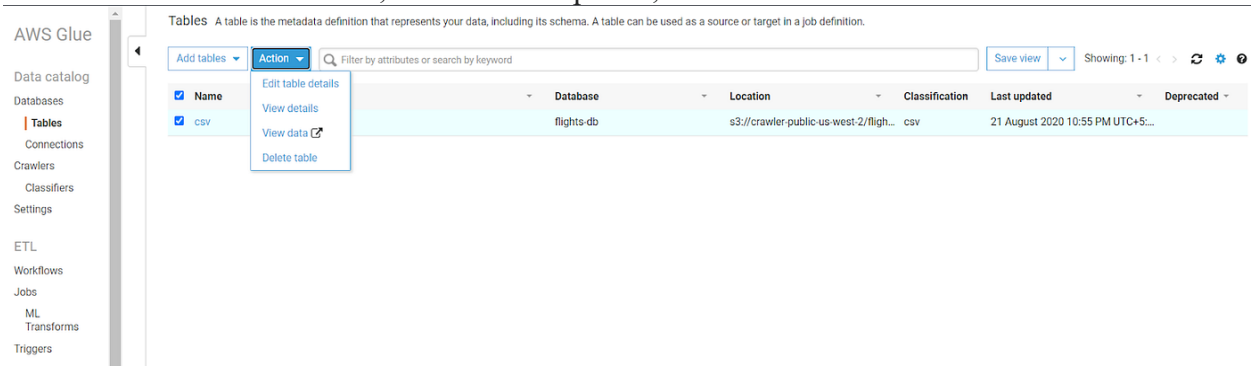
7. Click on the save job, and on the next page, you can see the flow diagram of the job and can edit the script generated.



8. You can use the inbuilt transform feature of AWS Glue to add some predefined transformation to the data.



9. Check the status of the Job, once it is completed, head out to the table section to see the data.



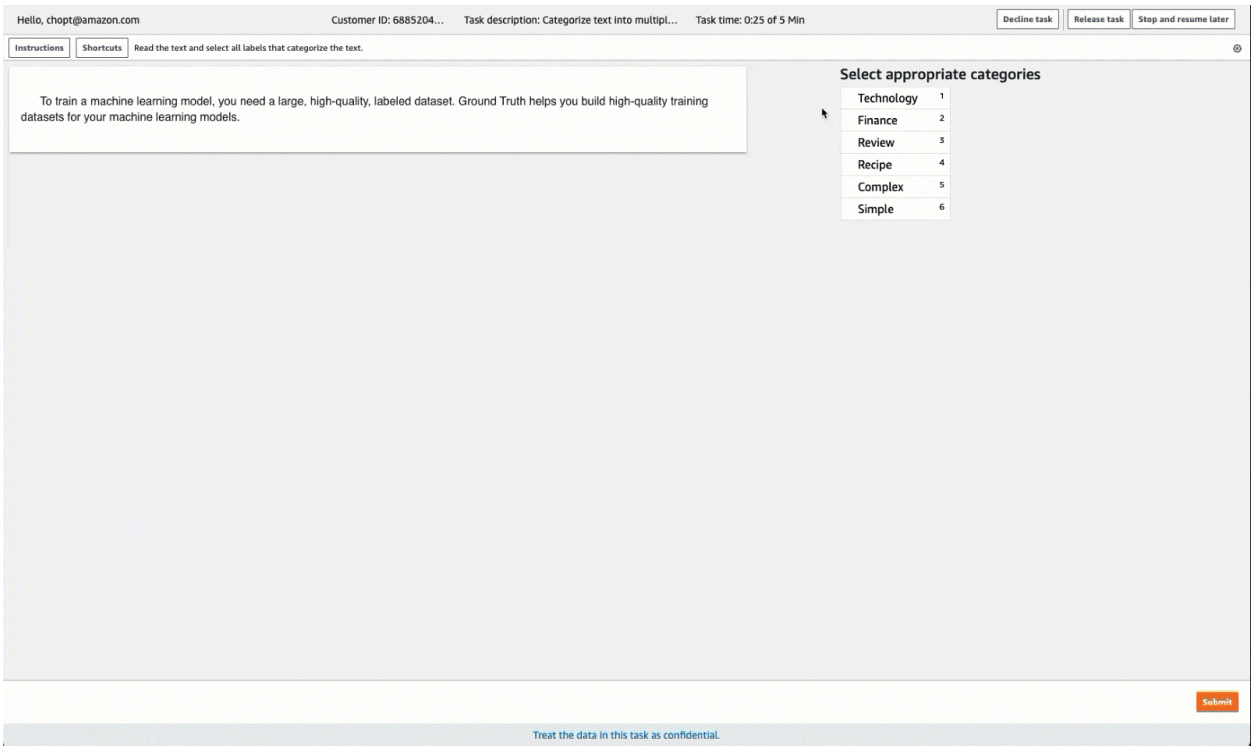
10. Click on View Data, it will open Athena, and preview a few records from the data.

(II) Perform Labelling:

Step1: Create a Multi-Label Text Classification Labeling Job (Console)

You can follow the instructions Create a Labeling Job (Console) to learn how to create a multi-label text classification labeling job in the Amazon SageMaker console. Choose Text from the Task category drop down menu, and choose Text Classification (Multi-label) as the task type.

Ground Truth provides a worker UI similar to the following for labeling tasks. When you create the labeling job with the console, you specify instructions to help workers complete the job labels that workers can choose from.



Step2: Multi-label Text Classification Output Data

Once you have created a multi-label text classification labeling job, your output data will be located in the **Output dataset location** field of the **Job overview** section of the console.

(III) Train and Build Model:

The following diagram shows how we train and deploy a model with Amazon SageMaker. Our training code accesses training data and outputs model artifacts from an S3 bucket. Then we can make requests to a model endpoint to run inference. We can store both the training and inference container images in an Amazon Elastic Container Registry (ECR).

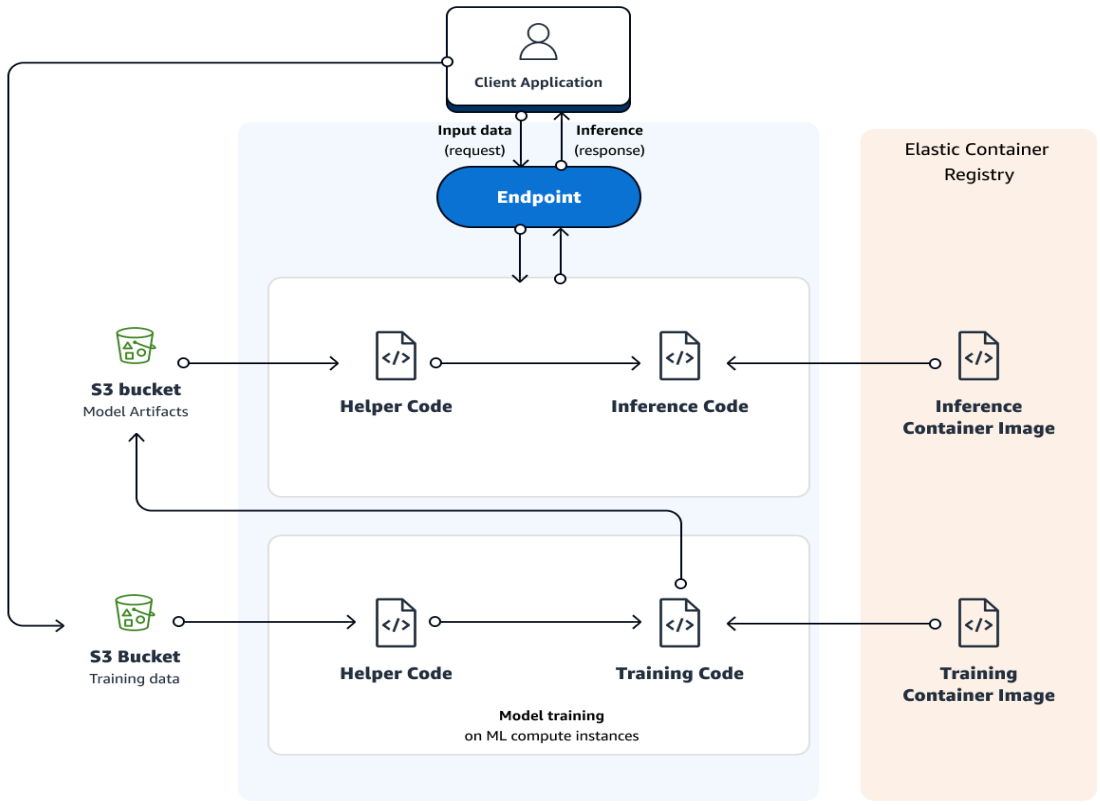


Fig. Train and deploy model through aws sagemaker

A.) To train a model in SageMaker, we create a training job. The training job includes the following information:

a. The URL of the Amazon Simple Storage Service (Amazon S3) bucket where we'll be storing the training data.
b. The compute resources that you want SageMaker to use for model training. Compute resources are machine learning (ML) compute instances that are managed by SageMaker.
c. The URL of the S3 bucket where you want to store the output of the job.
d. The Amazon Elastic Container Registry path where the training code is stored.

B.) **Submit custom code to train with deep learning frameworks**: we have option to submit custom Python code that uses TensorFlow, PyTorch, or Apache MXNet for model training.

C.) After you create the training job, SageMaker launches the ML compute instances and uses the training code and the training dataset to train the model. It saves the resulting model artifacts and other output in the S3 bucket you specified for that purpose.

D.) After training completes, SageMaker saves the resulting model artifacts to an Amazon S3 location that you specify.

(IV) Register The Model:

- You can register an Amazon SageMaker model by creating a model version that specifies the model group to which it belongs. A model version must include both the model artifacts (the trained weights of a model) and the inference code for the model.

- To create a Model Version in Amazon SageMaker Studio, complete the following steps.

- Sign in to Amazon SageMaker Studio. For more information, see Onboard to Amazon SageMaker Domain.

- In the left navigation pane, choose the **Home** icon ().
- Choose **Models**, and then **Model registry**.
- Open the **Register Version** form. You can do this in one of two ways:
- Choose **Actions**, and then choose **Create model version**.
- Select the name of the model group for which you want to create a model version, then choose **Create model version**.
- In the **Register model version** form, enter the following information:
- In the **Model package group name** dropdown, select the model group name.
- (Optional) Enter a description for your model version.
- In the **Model Approval Status** dropdown, select the version approval status.
- Enter your inference image location.
- Enter your model data artifacts location.
- (Optional) Provide details to aid endpoint recommendations.
- (Optional) Enter information about images preferred for transform and real-time inference jobs, and supported input and output MIME types.
- Choose **Register model version**.

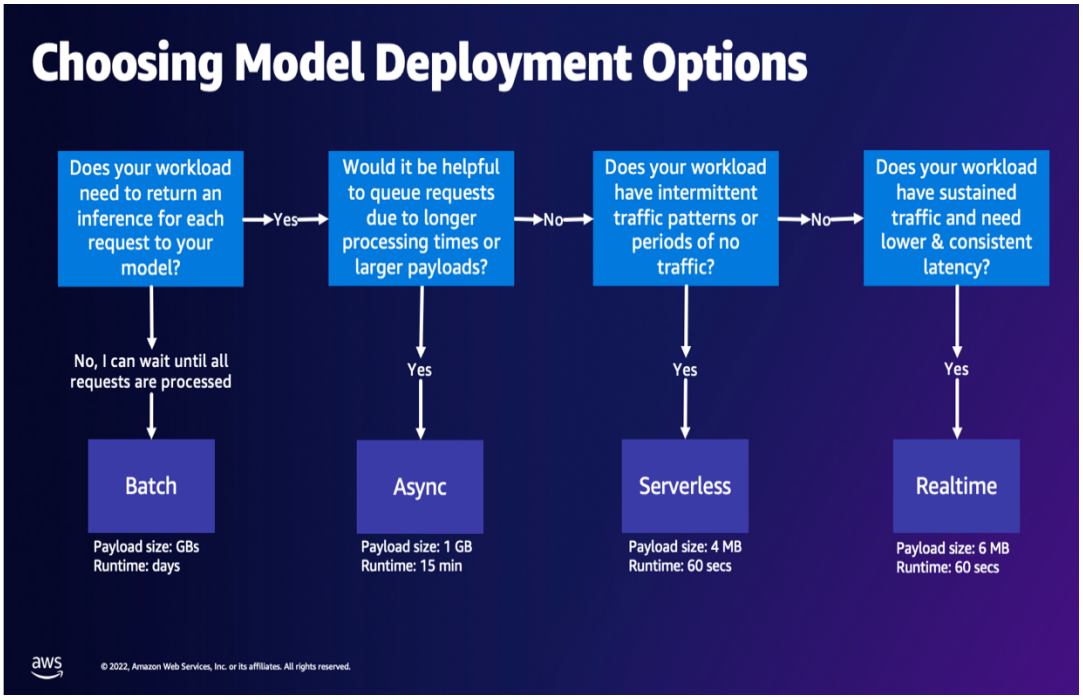
Deploy a Model from the Registry (SageMaker SDK)

- To deploy a model version using the Amazon SageMaker Python SDK use the following code snippet:

```
from sagemaker import ModelPackage
from time import gmtime, strftime

model_package_arn = 'arn:aws:sagemaker:us-east-2:12345678901:model-package/modeltest/1'
model = ModelPackage(role=role,
                     model_package_arn=model_package_arn,
                     sagemaker_session=sagemaker_session)
model.deploy(initial_instance_count=1, instance_type='ml.m5.xlarge')
```

(V) Real time serving:

Step 1: Choosing model deployment options.



Step 2: Creating an execution role for the REST API

Before you build the REST API, we need to create an execution role that gives your API the necessary permission to invoke your Amazon SageMaker endpoint. Complete the following steps:

1. On the IAM console, under Roles, choose Create Role.
2. For Select type of trusted entity, select AWS Service.
3. Choose the service API Gateway.
4. Choose Next until you reach the Review.

we can see that a policy has been included which allows API Gateway to push logs to CloudWatch.

5. Give your role a name, for example, APIGatewayAccessToSageMaker.
6. Choose Create Role.

we now need to add permissions so your role can invoke the Amazon SageMaker endpoint.

7. Use the role filter to find the role you just created.
8. Choose the role you created to see the role summary screen.
9. Choose Add Inline Policy.

10. On the Create Policy screen, select the service SageMaker and the action InvokeEndpoint.
11. For Resources, select Specific and enter the ARN for your Amazon SageMaker endpoint.
12. Choose Review Policy.
13. Give your policy a name, for example, SageMakerEndpointInvokeAccess.
14. Choose Create Policy.
15. On the role summary page, record the ARN for the role as you will need this later.

Step 3: Building an API Gateway endpoint.

In this section, you build your REST API.

Creating an API

Complete the following steps:

1. On the API Gateway console, choose Create API.
2. Choose REST.
3. Choose New API.
4. Give your API a name, for example, RatingsPredictor.
5. For Endpoint Type, choose Regional.
6. Choose Create API.

Note: In a production system, you should consider whether you would benefit from the Edge Optimized option.

Creating a resource

1. In the Resources editor, from the Actions menu, choose CreateResource.
2. For Resource Name, enter predicted-ratings.
3. Ensure that the Resource Path field also contains predicted-ratings.
4. Choose Create Resource.

You should see the predicted-ratings resource appear in the navigation tree.

5. Choose the predicted-ratings resource you created.
6. From the Actions menu, choose Create Resource.
7. For Resource Path, enter {user_id}.
8. For Resource Name, enter user_id.
9. Choose Create Resource.

Step 4: Deploying and testing the API

You are now ready to deploy your REST API. Complete the following steps:

1. On the Method Execution screen, under Actions, choose Deploy API.
2. For Deployment stage, choose [New Stage].

3. Give your stage a name, such as test.
4. Enter the required descriptions for this stage and deployment.
5. Choose Deploy.

You should now see a Stage Editor in the right-hand pane, with an Invoke URL endpoint for your deployed API. You can use curl to test your deployed endpoint.

23. FUTURE SCOPE

- **Voice assistants and chatbots:** With the growing popularity of voice assistants and chatbots, sentiment analysis can be integrated into these systems to understand and respond to user emotions and sentiments. This can enable more personalized and empathetic interactions, enhancing the user experience.
- **Product development and testing:** Companies can employ sentiment analysis to analyze user feedback during the product development lifecycle. By monitoring sentiments expressed in user surveys, feedback forms, and beta testing sessions, companies can identify potential issues, gather feature requests, and make data-driven decisions to improve their products.
- **Brand monitoring and reputation management:** Sentiment analysis can be used to monitor social media conversations, news articles, and online reviews to track public sentiment towards a company or brand. By monitoring sentiment in real-time, companies can quickly identify and address negative sentiment, manage their brand reputation, and take proactive measures to improve customer satisfaction.
- **Social listening and market research:** Sentiment analysis can be leveraged to gather insights from social media conversations and online forums related to specific products, services, or industry trends. This data can help companies understand consumer preferences, identify emerging trends, and adapt their strategies accordingly.
- **Customer experience optimization:** Sentiment analysis can be employed to analyze customer feedback across different touchpoints, such as surveys, call center interactions, and online reviews. Companies can identify patterns in customer sentiment, pinpoint areas for improvement, and enhance the overall customer experience.
- **Sentiment-based advertising and marketing:** Companies can use sentiment analysis to assess the sentiment of their target audience and tailor their advertising and marketing campaigns accordingly. By understanding the emotions and preferences of their audience, companies can create more relevant and impactful messaging.
- **Financial trading and investment:** Sentiment analysis can be applied to analyze news articles, social media posts, and other sources of market sentiment to support financial trading and investment decisions. By monitoring sentiment towards specific companies or sectors, traders and investors can gain insights into market sentiment trends and potentially make more informed decisions.
- **Health and well-being:** Sentiment analysis can play a role in monitoring mental health and well-being. Companies can develop sentiment analysis models to analyze social media posts, online support group discussions, or patient feedback to identify individuals who may require mental health support or intervention.

24. CONCLUSIONS:

- The period from April 8 to April 17 had the highest positive sentiment percentage, which might indicate a positive event or campaign during that time.
- May 4 to May 11 recorded the highest negative sentiment percentage, suggesting a possible incident or issue that led to a negative response.
- Tweets from the United States, India, and London, England are mostly neutral, while tweets from Lagos, Nigeria have a higher negative sentiment.
- The neural network model was trained for the future tweet prediction.
- Study on how model can be deployed on aws cloud platform and problem faced in making machine learning project end to end

25. REFERENCES:

1. https://www.kaggle.com/code/edomingo/etl-chatgpt-1000-daily-tweets-dataset
2. https://www.kaggle.com/code/vencerlanz09/chatgpt-tweets-visual-eda-and-sentiment-analysis#%F0%9F%94%ACOverview
3. https://github.com/hxycorn/Twitter-Sentiment-Analysis-about-ChatGPT
4. https://www.kaggle.com/code/shashidharnaiduboya/chatgpt-tweets-sentiment-analysis
5. https://www.kaggle.com/code/chadsaglam/chatgpd-tweets-with-deep-learning-using-bert
6. https://www.kaggle.com/code/larysakrasnova/analysis-content-of-responses/notebook
7. https://github.com/aws/amazon-sagemaker-examples/blob/main/introduction_to_amazon_algorithms/jumpstart_text_classification/Amazon_JumpStart_Text_Classification.ipynb
8. https://docs.aws.amazon.com/sagemaker/latest/dg/sms-text-classification-multilabel.html
9. https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_CreateTrainingJob.html
10. https://aws.amazon.com/blogs/machine-learning/part-2-model-hosting-patterns-in-amazon-sagemaker-getting-started-with-deploying-real-time-models-on-sagemaker/